



Critical Action Planning over Extreme-Scale Data

D4.1 Initial Report on Complex Event Forecasting, Learning and Analytics Version 1.0

Documentation Information

Contract Number	101092749
Project Website	https://crexdata.eu/
Contractual Deadline	M18, 06.2024
Dissemination Level	PU
Nature	R
Author	Elias Alevizos
Contributors	C Akasiadis, E Alevizos, G Anestis, A Artikis, D Banelas, A Deligiannakis, M Ebel, A B Eguzkitza, G Frangias, G Grigoropoulos, I Chamatidis, A Troupiotis- Kapeliaris, E. Filippou, N Giatrakos, M Juvillà, E Pitsikalis, P Mantenoglou, A Montagud, G Katsimpras, I Martinez, M Melero, T Ntiniakou, J Orama, G Paliouras, M Ponce de Leon, J Pottenbaum, V Samoladas, M Theologitis
Reviewer	Gennady Andrienko
Keywords	Complex event recognition, complex event forecasting, Bayesian optimization, optimization, cloud, edge, fog, data streams, federated machine learning, multilingual text mining, question answering, event detection, information extraction



CREXDATA has received funding from the European Union's Horizon Europe programme under grant agreement number 101092749.

D4.1 Initial Report on Complex Event Forecasting, Learning and Analytics
Version 1.0

Change Log

Version	Author	Date	Description Change
V0.1	All authors	29/05/2024	First complete version
V0.2	Elias Alevizos (NCSR)	30/05/2024	Merging, formatting, cleaning up
V0.3	Elias Alevizos (NCSR)	31/05/2024	Minor corrections before internal review
V0.4	All authors	17/6/2024	Incorporating comments from internal review
V0.9	Elias Alevizos (NCSR)	21/6/2024	Finalization
V1.0	Antonios Deligiannakis (TUC)	26/6/2024	Final version

Contents

Executive Summary	10
1 Introduction	12
1.1. Purpose and Scope	12
1.2. Relation to other Deliverables	12
1.3. Source code availability	12
1.4. Structure of the Deliverable	13
2 Multi-Resolution Complex Event Forecasting	14
2.1. Online optimization of Complex Event Forecasting	14
2.2. Complex Event Recognition with Symbolic Register Transducers	22
2.3. Complex Event Recognition with Allen Relations	28
2.4. Components for Critical Maritime Event Forecasting and Resolution	29
3 Interactive Learning for Simulation Exploration	36
3.1. Emergency Use-Case	36
3.2. Health Crisis Use-Case	41
3.3. Maritime Use-Case	46
3.4. Methods for the Exploration of Simulations Parameter Space	48
4 Federated Machine Learning	52
4.1. Related Work for Distributed & Federated Learning and Drawbacks	52
4.2. Relation to Use Cases	54
4.3. Functional Dynamic Averaging	54
4.4. Extending TensorFlow and KungFu to support FDA workflows	57
4.5. Experiments & Comparison to Prior Work	58
4.6. FDA in NeRF computation for the Weather Emergencies Use Case	66
5 Optimized Distributed Analytics as a Service	69

5.1. Motivation and Optimization Aspects in CREXDATA	69
5.2. Optimization Setup	72
5.3. Algorithmic Suite.....	74
5.4. Statistics Collection and the Need for Simulators.....	79
5.5. Experimentation.....	81
6 Text mining for Event Extraction	84
6.1. BERT Based Event Type Classifier	84
6.2. Question Answering for Event Information Extraction	89
7 Progress achieved towards the CREXDATA objectives.....	97
8 Acronyms and Abbreviations	98
9 References	99
10 Appendix 1	108
10.1. Appendix for Online Optimization of Complex Event Forecasting	108

List of Figures

Figure 1: Streaming symbolic automaton created from the speed related expression.	16
Figure 2: Offline CEF optimiser.	17
Figure 3: Architecture of RTCEF. Cylinders and rounded rectangles denote topics and services respectively. For simplicity, we omit synchronisation topics; instead we use grey arrows.	19
Figure 4: Experimental results for datasets MD{0/1/8} with Rport. 'rt' and 'opt' denote 'retrain' and 'optimisation'. Upper plots show MCC over time; lower plots show improvement.	21
Figure 5: Avg MCC (left-a, left-b) per FDi. Dataset and CER characteristics (right-a). 'v', 'vw/m' and 'm' stand for 'vessels', 'vessels with matches' and 'matches'. MPPT (right-b) per MDi for Rport.....	22
Figure 6 Throughput for sequential patterns with n-ary predicates as a function of pattern length. Window sizes are wstock = 500, wsmart = 5, wtaxi = 100.	26
Figure 7 Throughput for sequential patterns with n-ary predicates as a function of window size. Pattern length is 3.	27
Figure 8 Throughput for patterns with n-ary predicates and various operators. w = 1000.	27
Figure 9: Left: COLREG regions for vessel-to-vessel interaction classification [37]. Right: Optimal path planning and velocity adaption of a vessel with green being the final COLREG compliant optimal trajectory generated in each subsequent replanning step, black the COLREG compliant valid trajectories, and gray the invalid alternatives in each replanning step t. Image adapted from [36].....	30
Figure 10: Vessel collision avoidance workflow.....	30
Figure 11: Example of a head on collision and the generated COLREG compliant collision avoidance path.....	31

Figure 12: Visualization of the 2022 hurricane events (their paths) along the North Atlantic Ocean	32
Figure 13: a) Disruption of maritime traffic caused by Storm Ciarán that severely affected parts of Europe and the North Atlantic from late October to early November 2023. b) Comparison with the 6.11.2023, after the storm has calmed under normal weather conditions. Heatmap visualizes the wave height – direction. Different vessel types sailing in the area are visualized as triangles with colours indicating the different vessel type.	32
Figure 14: Mapping of weather features on the H3 index (example for NOAA wind dataset)	33
Figure 15: Download and fusion process of weather data from NOAA and Copernicus (Task groups).....	34
Figure 16: Statistics extraction and weather forecasts scale categorization process step for each weather parameter. Example showcases the task groups for days 0 (current) to 3 (Total task groups reach up to day 9)	34
Figure 17: Hazardous weather routing methodological workflow	35
Figure 18: Adoption of the simulator architecture [D2.1].....	37
Figure 19: Visualisation of an excerpt from the context data used for simulations	38
Figure 20: Output data from a sample simulation run (executed for a duration of 4 hours, in equal time steps of 60 seconds each) for total water depth in nodes (N11, N3, ...) and links (L10, L2, ...) in time steps 00:3:54:00 to 04:00:00 from res1d file and water level for grid elements in time steps 00:00:00 to 00:07:00 from dfsu/dfs2 file	39
Figure 21: (Left) A schematic of the interconnection between PhysiBoSS and Alya, the simulators, with the different components from the WP4 that will help with the calibration, the exploration and the interventions. (Right) Time-series example and illustration of the alternative interventions.	42

Figure 22: Model exploration workflow for the epidemiological scenario.....	43
Figure 23: Vessel collision avoidance interactive simulator	48
Figure 24: Preliminary results of the ETSC algorithms applied on the Multiscale Simulations CREXDATA sub-use-case.	50
Figure 25: LeNet-5 on MNIST	61
Figure 26: VGG16* on MNIST	62
Figure 27: DenseNet121 and DenseNet201 on CIFAR-10	62
Figure 28: Training accuracy progression with a (test) accuracy target of 0.8 (left), and 0.78 (right). A smaller final gap between training and target accuracy indicates less overfitting, i.e., better generalization capabilities of the trained model	63
Figure 29: VGG16* on MNIST – Accuracy Target: 0.994	63
Figure 30: DenseNet121 on CIFAR-10 – Accuracy Target: 0.8	64
Figure 31: DenseNet201 on CIFAR-10 – Accuracy Target: 0.78	64
Figure 32: Empirical Estimation of the Variance Threshold	65
Figure 33: ConvNeXtLarge on CIFAR-100 (transfer learning from ImageNet) — Deployment of FDA during the fine-tuning stage with Accuracy Target of 0.76	66
Figure 34: Communication time distribution with variable topology.....	66
Figure 35: Experimental setup	68
Figure 36: Logical Plan and Physical Network. The CREXDATA Optimized Analytics as a Service should device the placement of Logical Plan Operators to Physical Devices.....	70
Figure 37: Different physical plans prescribed as time passes from t_0 to t_{k+2}	71
Figure 38: Illustration of Pareto Optimal Physical Plans	72
Figure 39: Two Nodes of the Graph of Graphs representation of the CREXDATA streaming analysis as a service	73

Figure 40: Graph of graphs (GoG) as the search space encompassing all possible physical plans - workflows for a given set of workflows and a network setting. Each node in the figure includes a nested graph which is a physical plan/workflow. Edges correspond to simple, single actions (changes) leading from one physical workflow to another.	74
Figure 41: Illustration of the Greedy Search Algorithm Operation.....	77
Figure 42: Illustration of the Random Sampling Search Algorithm Operation	78
Figure 43: CREXDATA Simulator Executor and Statistics Collector significantly extending iFogSim.....	81
Figure 44: TRAIN workflow used in our experimental evaluation.....	82
Figure 45: Scalability Analysis and Parameter Tuning for the CREXDATA Algorithmic Suite (ESC, RSS, GSP Algorithms).....	83
Figure 46: Architecture.....	85
Figure 47 The system architecture of the QA model.....	91
Figure 48: Experimental results for datasets $FD_{\{0/3/5\}}$ with Rcards. 'rt' and 'opt' denote 'retrain' and 'optimisation'. Upper plots show MCC over time, while lower plots show improvement.	111
Figure 49: Avg MCC per FDi for Rcards.....	111

List of Tables

Table 1: An example stream composed of five events. Each event has a vessel identifier, a value for that vessel's speed and a timestamp.	16
Table 2: Overview of possible interactive learning scenarios for the EmCase.	40
Table 3: Epidemiological parameters of the MMCACovid19-vac infectious model.....	44
Table 4: Parameters to model the intervention to control the epidemic.....	45
Table 5: Interactive simulation scenario specifications for the Maritime Use Case components	46
Table 6: Setup.....	58
Table 7: Candidate simulators and their adoption/popularity, metrics collection, cloud to edge continuum coverage features	79
Table 8: Finetuning methods comparisons. The best performing result is indicated in bold. The second best is underlined for comparison.	88
Table 9: Comparison of base models with a finetuned model. The best performing result is indicated in bold. The second best is underlined for comparison.....	89
Table 10: Comparison of base models fine-tuned for event type prediction.....	90
Table 11: Results on the CrisisFACTS 2022 dataset. The best performing result is indicated in bold. The second-best is underlined for comparison.....	94
Table 12: Results on DL19 and DL20. The best performing result is indicated in bold. The second best is underlined for comparison.	95
Table 13: Results on BEIR. Best performing is marked bold. The second-best is underlined for comparison.	96

Executive Summary

The "CREXDATA Deliverable 4.1" report presents a comprehensive analysis and evaluation of the CREXDATA project, focusing on the development and implementation of advanced data processing techniques for real-time complex event processing. The project employs a distributed architecture using Kafka to facilitate communication between multiple services, ensuring efficient and scalable data handling. This architecture supports the integration and coordination of diverse data sources, enhancing the system's ability to process high volumes of data in real time.

Key contributions include the introduction of Symbolic Regular Expressions with Memory and Output (SREMO), which enhance the detection of complex relational patterns through nested operators and n-ary expressions. This innovative approach significantly improves the accuracy and efficiency of pattern recognition in real-time data streams, providing robust solutions for various applications. We also present a method for optimizing our Complex Event Forecasting engine in an online manner. This is a first step towards fine-tuning our engine for multi-resolution forecasting. It then presents an extension of our forecasting engine so that it can handle more expressive patterns.

With respect to critical maritime events, the document presents the components developed for forecasting and resolving vessel-to-vessel collisions and for hazardous weather rerouting at sea in the face of extreme weather events.

We then present the simulation scenarios from all three CREXDATA use-cases. Also, we provide an overview of the methods that we are going to apply for interactive simulation parameters exploration.

The report includes a detailed scalability analysis and parameter tuning of the algorithms used within the CREXDATA framework. This analysis underscores the importance of parallelization in optimizing the execution time of algorithms over networks composed of hundreds or thousands of devices. By effectively managing the computational load across multiple devices, the project ensures that the system can handle large-scale data processing tasks with high efficiency. This scalability is crucial for applications that require real-time processing of vast amounts of data, such as monitoring and responding to natural disasters.

Our work on Federated Machine Learning, presented next, focused on developing Functional Dynamic Averaging, a bandwidth-efficient technique for Federated Deep Learning. Comparison to previous state-of-the-art indicates orders-of-magnitude efficiencies in communication cost, especially in Computer Vision problems related to the Weather Emergency use case.

The algorithmic suite CREXDATA incorporates to support its overall architectural framework with optimized analytics as a service (i.e., attributing resources to the lot of CREXDATA extreme scale analytics workflows, on demand, simultaneously achieving good performance) is presented afterwards.

Next, we present a social media toolkit developed to monitor and extract key information from social networks to aid civil protection authorities during weather emergencies. This toolkit consists of a stream-enabled multilingual language model for event type detection, and a question answering model for extracting information from relevant social media posts. The report delves into the challenges and solutions associated with acquiring and mining useful information about emerging events from multilingual social media posts. These posts, often published by local government agencies or individuals directly involved in emergencies, provide valuable real-time perspectives. However, the unstructured and often ungrammatical nature of social media content, combined with its multilinguality, poses significant challenges. The project addresses these issues by developing advanced text mining algorithms that support civil protection authorities in responding to weather-induced emergencies, thus improving the overall efficiency and effectiveness of emergency response strategies.

Overall, the deliverable demonstrates substantial advancements in real-time data processing, contributing valuable methodologies and tools for emergency response and complex event detection. The CREXDATA project's innovations in distributed architectures, pattern recognition, and text mining represent significant strides in the field, offering practical solutions to real-world problems. This report not only highlights the technical achievements of the project but also emphasizes its potential impact on enhancing the responsiveness and effectiveness of emergency management systems.

1 Introduction

This document presents the progress of the CREXDATA project with respect to Complex Event Forecasting, Learning and Analytics. WP4 develops the forecasting, learning and scalability tools for analytics and distributed ML of CREXDATA. The tools developed in WP4 take into account the forecasting and scalability requirements that are specific to the use cases of WP2.

1.1 Purpose and Scope

The reader is expected to be familiar with Complex Event Forecasting, Artificial Intelligence, Federated Learning, Text Mining and Distributed processing techniques, as well as the general intent and concept of the CREXDATA project. The target readership is:

- CREXDATA researchers
- CREXDATA audit

CREXDATA focuses on event forecasting, interactive and federated machine learning and text mining techniques for large scale data. This document presents the current advancements and discusses the scientific and technological issues that are being investigated in Work-Package 4, with respect to Complex Event Forecasting, Learning and Analytics.

1.2 Relation to other Deliverables

This document is related to the following project deliverables:

- D2.2 Initial Use Case Evaluation, Pilots, Demonstrators and Simulation Models and Tools;
- D3.1 Initial Report on System Architecture, Integration and Released Software Stacks.

1.3 Source code availability

CREXDATA has promised to deliver the source code of its solutions. Below we provide a list of source code repositories related to the present deliverable. Please note that some of the repositories may be anonymous because the relevant papers have not been published yet.

- Repositories for Task 4.1
 - <https://github.com/EIAlev/Wayeb>
 - <https://github.com/manospits/rtcef>
 - <https://github.com/EIAlev/cer-srt>
- Repositories for Task 4.2
 - <https://github.com/xarakas/ETSC>
- Repositories for Task 4.3

- <https://github.com/miketheologitis/FedL-Sync-FDA>
 - <https://github.com/miketheologitis/TFD-FedL-Sync-FDA>
 - <https://github.com/gfrangias/KungFu>
- Repositories for Task 4.4
 - <https://github.com/DBanelas/crexdata-optimizer>
 - <https://github.com/DBanelas/placement-simulation-suite>
- Repositories for Task 4.5
 - <https://github.com/langtech-bsc/crexdata-task4.5>
 - <https://anonymous.4open.science/r/genra-BB1B>

1.4 Structure of the Deliverable

This document has the following structure:

- Section 2 presents a method for optimizing our Complex Event Forecasting engine in an online manner. This is a first step towards fine-tuning our engine for multi-resolution forecasting. It then presents an extension of our forecasting engine so that it can handle more expressive patterns and our work of extending an event recognition engine with complex temporal relations. Finally, the components for forecasting and resolving critical maritime events are presented.
- Section 3 presents the simulation scenarios from all three CREXDATA use-cases. Also, we provide an overview of the methods that we are going to apply for interactive simulation parameters exploration, such as active learning, optimization of interventions, early time-series classification, and reinforcement learning.
- Section 4 presents our work on Federated Machine Learning. It focuses on developing Functional Dynamic Averaging, a bandwidth-efficient technique for Federated Deep Learning. Comparison to previous state-of-the-art indicates orders-of-magnitude efficiencies in communication cost, especially in Computer Vision problems related to the Weather Emergency use case.
- Section 5 presents the CREXDATA optimization approach which automates the process of mapping logical workflows (workflows which only describe the application logic, being deprived of any physical execution aspect) to physical workflows deployable across the networked settings over which CREXDATA operates. It details the optimization algorithmic suite of CREXDATA and explains the way it optimizes the execution of arbitrarily many workflows over arbitrarily many devices, under arbitrarily many physical execution options in volatile streaming and network settings.
- Section 6 presents the text mining algorithms and language models developed for monitoring and extracting key information about weather emergencies from social media messages. It details the language model for event type detection, the module for information extraction, and the design decisions in each step.

2 Multi-Resolution Complex Event Forecasting

In this Section, we present our work for multi-resolution Complex Event Forecasting (CEF). Our work is based on an already existing forecasting engine which we have developed in previous projects, called Wayeb¹ [1] [2]. Wayeb is a forecasting engine which employs symbolic automata as its computational model. Wayeb is both efficient and expressive, while maintaining clear, compositional semantics for the patterns expressed in its language due to the fact that symbolic automata have nice closure properties. At the same time, it is expressive enough to support most of the common Complex Event Recognition operators. Specifically, our contributions for the first half of the project are the following:

- Section 2.1 presents *RTCEF*, an open-source novel framework for run-time optimisation of CEF. More specifically, *RTCEF*, aims to facilitate online CEF training/hyperparameter optimization over streams with constantly evolving conditions. We evaluate *RTCEF* on two real-world use-cases from the maritime and financial domains and our reproducible results show that *RTCEF* can significantly improve forecasting performance with minimal lag upon run-time changes. For the second half of the project, the goal is to extend this optimization technique in order to target forecasting at multiple temporal resolution / earliness values.
- Section 2.2 presents an extension of Wayeb which allows it to handle more expressive patterns, required for the CREXDATA project. Wayeb can now accommodate patterns with relational constraints, e.g., a decreasing trend in the number of COVID cases. We present a summary of our results in order to make the deliverable succinct. A complete description of our work may be found in an extended technical report².
- Section 2.3 presents an extension of another event recognition engine that we have at our disposal, RTEC³. The engine is extended so as to be able to handle relations expressed in Allen's interval algebra, thus significantly increasing its expressive power.

2.1 Online optimization of Complex Event Forecasting

CEF, among other reasoning tasks, operates over constantly evolving conditions. Take for example the problem of maritime route optimisation. Vessels may follow a different route depending on the weather conditions or in general the season of the year [3]. Another example is financial fraud detection—fraudsters constantly adapt their tactics to avoid getting caught. While such problems can be treated by offline trained models for a given time period, in practice, such models fall short in settings where dynamic changes that invalidate previous training data are present. CEF systems in particular, typically, rely on probabilistic models trained on historical data [4] [5] [1]. This renders CEF systems inherently susceptible to evolutions in the input that can invalidate their underlying models—recall the example mentioned earlier relating maritime routes with weather. Additionally, as with the majority of

¹ <https://github.com/EIAlev/Wayeb>

² <https://github.com/EIAlev/cef-srt/blob/main/docs/cef-srt-extended-report.pdf>

³ <https://github.com/aartikis/RTEC>

trainable models, CEF models have multiple hyper-parameters that require fine tuning for optimal performance. Wayeb [2], one of the first CEF engines, is no exception to the above. In prior work [6], a methodology for hyper-parameter optimisation, specifically tailored for Wayeb, was proposed. While this method can successfully find near-optimal hyper-parameters in the offline setting, it cannot handle dynamic evolutions of the input that can, in the future, deteriorate CEF performance.

To address the above challenges, we propose *RTCEF* a novel framework for Run-Time optimisation of CEF. *RTCEF*⁴, adopts a distributed architecture comprising several services communicating via Kafka [7] [8] [9], and allows run-time update of CEF models. In other words, it supports continuous adaptation to dynamic changes in the input stream while also ensuring little to no downtimes in CEF. Since hyper-parameter optimisation is an expensive procedure, *RTCEF* employs a trend-based policy which allows, in addition to hyper-parameter tuning, CEF re-training without changing hyper-parameters. The contributions of this work are thus the following: (a) we introduce *RTCEF*, an open-source novel framework for run-time optimisation of CEF; (b) the distributed architecture we employ allows CEF to run in parallel to training, or optimisation tasks, therefore ensuring no disruptions; (c) we extensively evaluate *RTCEF* on two real-world use-cases from the maritime and financial domains; (d) our reproducible results show that *RTCEF* can significantly improve forecasting performance with minimal lag upon run-time changes.

2.1.1 Background

Complex Event Forecasting

CEF is a task that allows forecasting CEs of interest over an input stream of low level events; e.g., timestamped position messages of maritime vessels, or credit card transactions. Forecasts involve the occurrence of a CE in the future accompanied by a degree of certainty [2]. This behaviour is usually derived from stochastic models that project into the future evolutions of the input that can cause a detection of a CE. For the task of CEF, we utilise Wayeb, a CEF engine which employs symbolic automata as its computational model. Wayeb has clear, compositional semantics for the patterns expressed in its language and can support most of the common operators [10]. Wayeb's patterns are expressed as Symbolic Regular Expressions (SREs), where terminal expressions are Boolean expressions, i.e. logical formulae that use the standard Boolean connectives. Formally, Wayeb SRE(s) are defined using the grammar below:

$$R ::= R_1 + R_2 (\text{union}) \mid R_1 \cdot R_2 (\text{concatenation}) \mid R_1^* (\text{Kleene-star}) \\ \mid !R_1 (\text{complement}) \mid \psi (\text{Boolean Expression})$$

where R_1, R_2 are also regular expressions, and ψ is a Boolean expression. The semantics of the above operators are detailed in [2]. Evaluation of SRE on a stream of events requires first their compilation into symbolic automata. Transitions in symbolic automata are labelled with Boolean expressions. For a symbolic automaton to move to another state, it first applies the Boolean expressions of its current state's outgoing transitions to the element last read from the stream. If an expression is satisfied, then the corresponding transition is triggered and the automaton moves to that transition's target state. For example, in a maritime monitoring scenario, a domain expert could use Wayeb's language to specify a pattern $R := (\text{speed} > 10) \cdot (\text{speed} > 10)$ for identifying speed violations in specific areas where the

⁴ <https://github.com/manospits/rtcef>

maximum allowed speed is 10 knots. This pattern is satisfied when there are two consecutive events where a vessel's speed exceeds the threshold. We require two consecutive violations in order to avoid situations where the vessel has only a momentary or random lapse.

Table 1: An example stream composed of five events. Each event has a vessel identifier, a value for that vessel's speed and a timestamp

vessel id	78986	78986	78986	78986	78986	...
speed	5	3	9	14	11	...
timestamp	1	2	3	4	5	...

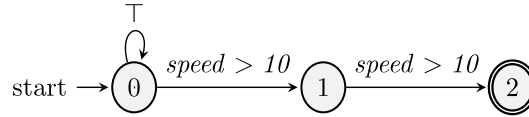


Figure 1: Streaming symbolic automaton created from the speed related expression

The compiled automaton corresponding to R is illustrated in **Figure 1**. For an input stream consisting of the events in **Table 1** the automaton would run as follows. For the first three input events, the automaton remains in state 0. After the fourth event, it moves to state 1 and after the fifth event it reaches its final state, state 2, triggering also a CE detection for R at timestamp = 5.

To perform CEF, we need a probabilistic description for a symbolic automaton derived from a SRE. For this purpose, we employ Prediction Suffix Trees (PSTs) [11] [12]—a form of Variable-order Markov Models (VMM). VMMs, compared to fixed-order Markov models, capture longer-term dependencies as in practice they allow for higher order (m) values than the latter. Each node in a PST, contains a “context” and a distribution that indicates the probability of encountering a symbol, conditioned on the context. **Figure 2** (top left) shows an example of a PST. Each “symbol” of a PST corresponds to a predicate of the automaton for which we want to build a probabilistic model. Consequently, with the use of a PST, for every state q of an automaton, we can calculate the waiting-time distribution (W_q), that is, the probability of reaching a final state in n transitions from state q . Recall that a CE is detected whenever an automaton reaches a final state. **Figure 2** (middle and bottom left) shows an example of an automaton and the waiting-time distributions learned from a training dataset. Taking all of the above into account, Wayeb performs CEF as follows. Given the current state q of an automaton, using W_q , it computes the probability of reaching a final state (p_{CE}) within the next n transitions (or, equivalently, input events). If p_{CE} exceeds a confidence threshold $\theta_{fc} \in [0,1]$, Wayeb emits a “positive” forecast (denoting that the CE is expected to occur), otherwise a “negative forecast” (no CE is expected) is emitted.

The above discussion illustrates the need for optimising multiple hyper-parameters. For example, although setting the maximum order m of the VMM generally improves accuracy, it leads to longer training times. Similarly, finding the optimal value for θ_{fc} is a crucial step as low or high θ_{fc} values can cause many false positives or false negatives respectively. Manually fixing the above parameters would lead to severely sub-optimal results. On the contrary, exhaustive parameter space exploration is of high computational complexity making it prohibitive for run-time settings. To address these issues, we employ Bayesian

optimisation to efficiently explore only a small fraction of the parameter space and learn the optimal hyper-parameter values for the entire parameter space.

Offline Optimisation

In prior work [6] a framework for offline hyper-parameter optimisation of CEF was introduced. The offline CEF optimiser of [6] and RTCEF utilise Bayesian optimisation as the underlying mechanism for hyper-parameter calibration. Bayesian Optimisation (BO) [13] [14] is a stochastic method used for optimising costly objective functions that are complex or unknown. In our context, the objective function describes the performance of a CEF system. Therefore, the goal of BO is to find the vector of system parameters that maximises CEF performance, using a targeted, minimal set of system runs, termed ‘micro-benchmarks’ (essentially function evaluations), as training samples. We want to achieve the best possible CEF performance using as few micro-benchmarks as possible; this is a setting where BO perfectly fits [13] [14] [6]. BO employs a surrogate model, usually a Gaussian Process (GP) Regressor (GPR), to approximate the objective function and iteratively refines this model. Priors about the objective function, often for the mean and covariance functions of the GP model, are set before any data observation. Every time we observe a new micro-benchmark and collect system performance metrics, we acquire a training sample to fit on the GPR, thereby updating our posterior belief in light on new evidence. In this work, the posterior distribution represents our updated knowledge about the CEF system's performance. The posterior distribution after observing n new training samples (i.e., system runs in our case) is given by the surrogate model that has been updated with the newly acquired knowledge about CEF performance under a new hyper-parameter combination using Bayesian inference (see for example [6]).

Micro-benchmark selection starts with an initial set chosen randomly from the input parameter domain; we execute respective micro-benchmarks and observe performance metrics in a set D_{init} . Subsequent micro-benchmarks are selected by an acquisition function (e.g., Expected Improvement) which balances exploration of unexplored regions and exploitation of current knowledge to identify points likely to yield the best performance. For instance, in the plot at the bottom right of **Figure 2**, the acquisition function chooses the point in the input domain with the highest uncertainty. BO concludes either when a micro-benchmark budget is depleted or when the optimal value for the objective function converges. The plot at the top right of **Figure 2** illustrates a GPR with minimal uncertainty around its mean values, after the microbenchmark budget has been fitted.

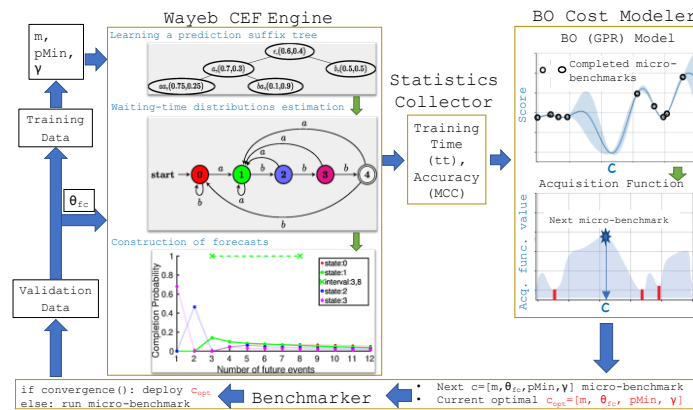


Figure 2: Offline CEF optimiser

The work of [6] introduces the offline CEF optimiser, a framework for offline hyper-parameter optimisation of CEF. **Figure 2** illustrates the offline CEF Optimiser [6], comprising the CEF Engine alongside Benchmarkers, Statistics Collector, and BO Cost Modeller. The CEF Engine utilises training and validation datasets to construct probabilistic models for Wayeb, while the remaining components execute BO (offline), as described earlier. The Benchmarkers initialise optimisation by sampling configurations and conducting micro-benchmarks, while the Statistics Collector gathers performance statistics for the BO Cost Modeller. In this setting, the relevant performance metrics are Matthew's Correlation Coefficient (MCC) and training time (tt) combined together in a single formula, i.e. $\text{Score}(c) = w_1 \times MCC(c) - w_2 \times \tanh\left(\frac{tt(c)}{\theta_{tt}} - 1\right)$, where θ_{tt} is a desired, target training time above which tt values get penalised. $w_1 + w_2 = 1$ are weights that allow adjustable emphasis on MCC or tt . The BO Cost Modeller fits a GPR to D_{init} and prescribes subsequent micro-benchmarks using an acquisition function. The Benchmarkers decide whether optimisation should conclude or continue based on BO convergence. In the former case, the optimal configuration is deployed, while in the latter further micro-benchmarks are conducted. This iterative process ensures the deployment of the most effective CEF Engine configuration while maximising performance and minimising computational overhead.

On the other hand, the offline CEF optimiser, suffers from several disadvantages: (i) it drives its decisions by attributing equal importance to all performance metrics, while in a streaming setup we often need to take into consideration only a sliding window of recent measurements and defy obsolete ones; (ii) it cannot optimise CEF hyper-parameters at run-time which is a crucial limitation, since fluctuations in the input's statistical properties in streaming settings is the norm rather than an infrequent situation; (iii) it cannot distinguish whether model hyper-parameters should be adjusted due to such statistical changes or if it is only the Wayeb's internal probabilistic model (PST) that should be re-trained, without hyper-parameter re-calibration. *RTCEF*, presented in the next section, addresses all these issues.

2.1.2 Run-time CEF optimisation

RTCEF, is built with two major goals in mind. First it should allow run-time updating of CEF models for treating input data evolutions that would otherwise lead to deterioration of forecasting performance; and second, it should perform CEF with no disruptions, i.e. model updating should not cause delays on CEF. Here, we present the architecture of our framework from a bird's eye perspective while in Appendix 10.1.1 we describe each service with more details.

Architectural overview

The architecture of *RTCEF* consists of five main services, acting as Kafka producers and consumers, running synergistically to ensure undisrupted CEF as well as dynamic CEF model retraining or optimisation. **Figure 3** illustrates these services and the communication lines between them. *RTCEF* is divided into four main parts, namely CEF, data collection, monitoring of scores, and optimisation or re-training. Below we describe these parts.

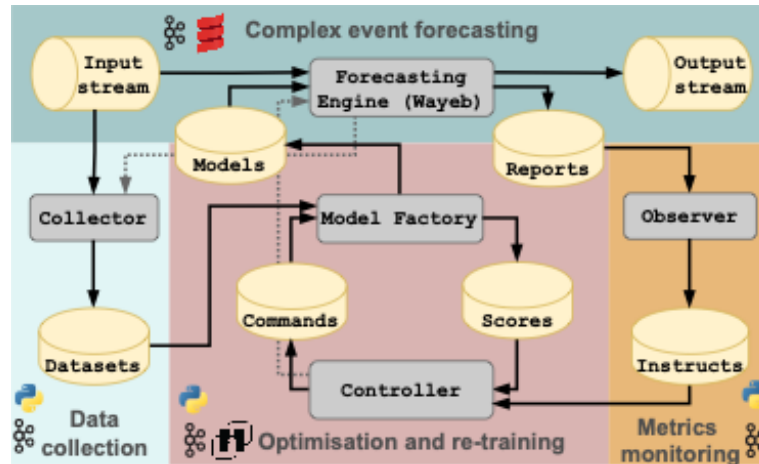


Figure 3: Architecture of RTCEF. Cylinders and rounded rectangles denote topics and services respectively. For simplicity, we omit synchronisation topics; instead we use grey arrows

CEF. The CEF part contains the forecasting engine service, which in our case is Wayeb. Wayeb reads timestamped simple events from the input stream, and produces an output stream of CE forecasts. In parallel, the engine reads from the models topic, which contains updated model versions, i.e. PSTs. The engine runs undisrupted, pausing momentarily only for model replacement when needed as explained shortly. Finally, the engine additionally produces a stream of CEF forecasting performance reports. Recall that Wayeb performs both CEF and CER, therefore scores can be produced on-the-fly; every forecast can be evaluated by the presence (or absence) of subsequent CEs.

Data collection. This part of RTCEF handles data collection from the input stream (e.g., a set of maritime vessel positional messages or credit card transactions). Data collection is needed as retraining or optimisation procedures require training datasets. In the run-time setting training datasets evolve over time. Therefore, the data collection part of RTCEF includes the collector service, a data processing module, that organises and stores, according to some policy, subsets of the input stream. This ensures that up-to-date datasets are available for subsequent re-training or optimisation.

Performance monitoring. In order to determine whether the performance of the CEF engine (i.e., the *MCC* score) has deteriorated, the quality of its forecasts must be monitored. This task is handled by the observer service which consumes CEF performance reports from the ‘reports’ topic, and produces ‘retrain’ or ‘optimise’ instructions via a trend-based policy.

Optimisation and re-training. The final part of RTCEF involves two services, a model factory service, and a controller service. The controller reads the instructions (‘retrain’ or ‘optimise’) of the observer, and accordingly initiates a training or optimisation procedure—we will refer to both as ‘update model procedure’. For ‘retrain’ instructions, the controller sends a ‘train’ request to the model factory for producing a model. Respectively, for ‘optimisation’ instructions, the controller initiates an optimisation message exchange with the factory, whereby the controller sends ‘train & test’ requests, while the factory replies with performance metrics. In both ‘train’ and ‘train & test’ requests, the controller service supplies the model hyper-parameters. Note that during an update model procedure, the factory will

use the latest dataset made available by the collector. When a new model procedure is completed, the model factory sends a new model version to the ‘Models’ topic.

2.1.3 Experimental evaluation

Experimental setup

We evaluate RTCEF on maritime situational awareness and credit card fraud management. For a smoother presentation we will omit experiments concerning credit card fraud management from here and present them later in Appendix (Section 10.1).

Maritime Situational Awareness. We use a real-world, publicly available, maritime dataset containing 18M spatio-temporal positional AIS (Automatic Identification System) messages transmitted between October 1st 2016 and 31st March 2026 (6 months), from 5K vessels sailing in the Atlantic Ocean around the port of Brest, France [15]. AIS allows the transmission of information such as the current speed, heading and coordinates of vessels, as well as, ancillary static information such as destination and ship type. We evaluate RTCEF on a maritime pattern, which expresses the arrival of a vessel at the main port of Brest [6]:

$$R_{\text{port}} := (\neg \text{InsidePort}(\text{Brest})) * \cdot (\neg \text{InsidePort}(\text{Brest})) \cdot \\ (\neg \text{InsidePort}(\text{Brest})) \cdot (\text{InsidePort}(\text{Brest}))$$

InsidePort(Brest) is true, when a vessel is within 5 km from the port of Brest. Consequently, R_{port} is satisfied if a sequence of at least three events occur. The first two require the vessel to be away from the port—thus limiting false positives from noisy entrances—, while the last denotes that the vessel has entered the port. This CE is important for port management and logistics reasons. Furthermore, we perform experiments for a CE termed R_{fish} , and satisfied when vessels enter fishing areas and sail with fishing speed. To cross validate our approach, we create 6 datasets ($MD_i, i \in [0,5]$) by shifting the starting month in a cyclic manner.

RTCEF initialisation. RTCEF requires an initial forecasting model for the engine. Alongside this model, the hyper-parameters used for its creation must be provided to the controller, ensuring their availability for any ‘retrain’ commands. Furthermore, if the initial model was produced via offline BO (as in [6]) then a sample of its micro-benchmarks can be supplied to the controller service. If prior samples are not available, then the first ‘optimisation’ call starts from scratch. Finally, the user must provide a configuration file whereby parameters such as bucket_size and max_slope are set (see Appendix, Section 10.1).

We perform offline hyper-parameter optimisation on the first four weeks of each dataset MD_i and use the resulting model, hyper-parameters and micro-benchmark samples for initialisation. We set w_2 in Score(c) (see Background) to 0 as we focus on improving MCC scores. Since model update procedures happen in parallel to CEF, training time is no longer of importance. Concerning the hyper-parameter space, we chose the same setting as in [6]. To showcase, the benefit of RTCEF, we additionally perform experiments with the offline CEF optimiser (see Background): i.e. for each FD_i/MD_i we perform CEF using the initial model of each dataset. In what follows, the experiments that utilise our framework are labelled as ‘fr’ while experiments that are performed only with offline trained models are labelled as ‘no fr’.

RTCEF is implemented in Python 3.9.18, while the used Kafka version was 3.5.2. For BO, we use the scikit-optimize library [16] 0.9.0. The experiments are conducted on a server

running Debian 12 with an AMD EPYC 7543 32-Core Processor and 400G of RAM. Our framework is open-source and our experiments are fully reproducible.

Experimental results

Figure 4 shows the evolution of MCC over time for R_{port} along with the score improvements for the cases ‘fr’ and ‘no fr’ for the maritime $MD_{0/3/5}$ datasets. Results concerning MD_0 —the dataset in its original order—show that the offline approach (‘no fr’) demonstrates poor performance and significant fluctuations in MCC scores over time. Our approach improves scores and reduces fluctuations.

For MD_0 RTCEF dramatically improves MCC up to $\sim 300\%$ following retraining and optimisation procedures in weeks 5 and 6, respectively. A similar pattern is observed on the MD_5 dataset. On the MD_3 case, results show that the initial model, generated by hyperparameter optimisation on weeks 12 to 15, underperforms. However, this behaviour is immediately cured when the observer requests optimisation in the first running week (see orange dot in week 16 of **Figure 4** middle)—this is due to the score being less than min_score (see **Algorithm**). We attribute the low scores of the initial model of the MD_3 dataset on the lack of vessels passing through the monitoring area on that period (see **Figure 5** right-a). **Figure 5** left-a, shows that the average MCC for each dataset MD_i ($i \in [0,5]$) when using RTCEF (‘fr’) is consistently higher than that achieved via a single model trained only on the first four weeks of each dataset (‘no fr’). In **Figure 5** left-b we report results concerning the R_{fish} CE. For the R_{fish} pattern there are no input data evolutions that affect CEF performance, therefore in this case, the results show that when data evolutions that affect model performance are not present, using RTCEF does not affect forecasting performance.

Concerning processing efficiency, interruptions in CEF are minimal as retraining or optimisation procedures occur in parallel to CEF, thus efficiency remains unaffected. However, when a new model request arises, new model versions arrive with some delay. Recall, that until a new model is available, the engine consumes, in parallel, the input stream with the already deployed model. **Figure 5** right-b shows the mean percentage of time spent every four weeks for production of models (we denote this value as $MPPT$) involving the R_{port} pattern. The results show that every four weeks, on average less than $\sim 0.2\%$ of time is spent for model production (roughly 70 minutes). Consequently, RTCEF spends minimal time every four weeks for model production, thus ensuring minimal delays and a resource-friendly behaviour as optimisation or retraining procedures are not overperformed.

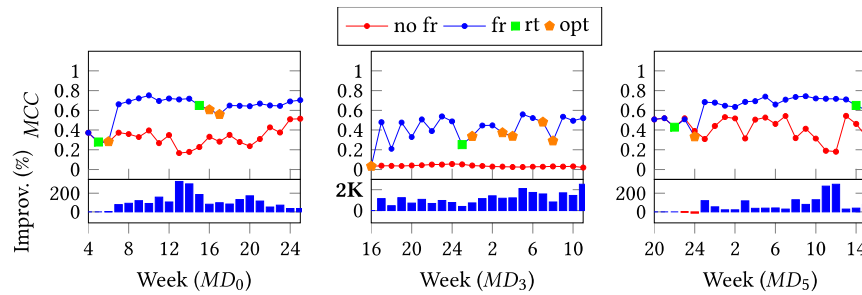


Figure 4: Experimental results for datasets $MD_{0/1/8}$ with R_{port} . ‘rt’ and ‘opt’ denote ‘retrain’ and ‘optimisation’. Upper plots show MCC over time; lower plots show improvement

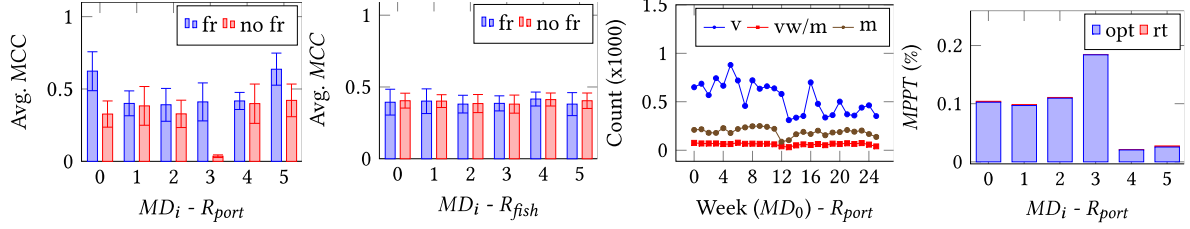


Figure 5: Avg MCC (left-a, left-b) per FDi. Dataset and CER characteristics (right-a). ‘v’, ‘vw/m’ and ‘m’ stand for ‘vessels’, ‘vessels with matches’ and ‘matches’. MPPT (right-b) per MDi for Rport

2.2 Complex Event Recognition with Symbolic Register Transducers

Automata are of particular interest for the field of CER, because they provide a natural way of handling sequences. As a result, the usual operators of regular expressions, like concatenation, union and Kleene-star, have often been given an implicit temporal interpretation in CER. For example, the concatenation of two events is said to occur whenever the second event is read by an automaton after the first one, i.e., whenever the timestamp of the second event is greater than the timestamp of the first. On the other hand, atemporal constraints are not easy to define using classical automata, since they either work without memory or, even if they do include a memory structure, e.g., as with push-down automata, they can only work with a finite alphabet of input symbols. For this reason, the CER community has proposed several extensions of classical automata. These extended automata have the ability to store input events and later retrieve them in order to evaluate whether a constraint is satisfied [17] [18] [19]. They resemble both register automata [20], through their ability to store events, and symbolic automata [21], through the use of predicates on their transitions. They differ from symbolic automata in that predicates apply to multiple events, retrieved from the memory structure that holds previous events. They differ from register automata in that predicates may be more complex than that of (in)equality.

One issue with these CER-specific automata is that their properties have not been systematically investigated, in contrast to models derived directly from the field of languages and automata; see [22] for a discussion about the weaknesses of automaton models in CER. Moreover, they sometimes need to impose restrictions on the use of regular expression operators in a pattern, e.g., nesting of Kleene-star operators is not allowed. We propose a system for CER, based on an automaton model which can address these issues. This model is a combination of symbolic and register automata. We call such automata Symbolic Register Transducers (SRT). SRT extend the expressive power of symbolic and register automata, by allowing for more complex patterns to be defined and detected on a stream of events. We also present a language with which we can define patterns for complex events that can then be translated to SRT. We call such patterns Symbolic Regular Expressions with Memory and Output (SREMO), as an extension of the work presented in [23], where Regular Expressions with Memory (REM) are defined and investigated. \rem\ are extensions of classical regular expressions with which some of the terminal symbols of an expression can be stored and later be compared for (in)equality. SREMO allow for more complex

conditions to be used, besides those of (in)equality. They additionally allow each terminal sub-expression to mark an element as belonging or not to the string/match that is to be recognized, thus acting as transducers.

Our contributions may then be summarized as follows:

- We present a CER system based on a formal framework with denotational and compositional semantics, where patterns may be written as Symbolic Regular Expressions with Memory and Output (SREMO).
- We show how this framework subsumes, in terms of expressive power, previous similar attempts. It allows for nesting operators and selection strategies. It also allows n-ary expressions to be used as conditions in patterns, thus allowing the detection of relational patterns.
- We extend previous work on automata and present a computational model for patterns written in SREMO, Symbolic Register Transducers (SRT), whose main feature is that it supports relations between multiple events in a pattern. SRT also have the ability to mark exactly those simple events comprising a complex one.
- We show that SRT are closed under the most common operators, i.e., union, intersection, concatenation and Kleene-star. Moreover, we show that, by using windows, SRT are closed under complement and determinization. Windows are an indispensable operator in CER because, among others, they limit the search space for pattern matching.
- We implement a CER engine with SRT at its core and present relevant experimental results. Our engine is both more efficient than other engines and supports a language that is more expressive than that of other systems.

2.2.1 Symbolic Register Transducers

We start by presenting a language for CER and discuss its semantics. The main feature of this language is that it allows for most of the common CER operators (such as selection, sequence, disjunction and iteration), without imposing restrictions on how they may be used and nested. Our proposed language can also accommodate n-ary conditions, i.e., we can impose constraints on the patterns which relate multiple events of a stream, e.g., that the number of tumour cells at the current timepoint is higher than their number at the previous timepoint. We also discuss the semantics of patterns written in the proposed language and show that these are well-defined. Hence, in order to know whether a given stream contains complex events corresponding to a given pattern, we do not need to resort to a procedural computational model. The semantics of the language may be studied independently of the chosen computational model. This feature is critical, as it allows for a systematic understanding of the use of operators. Additionally, it could be of importance for optimization, which often relies on pattern re-writing, assuming that we can know when two patterns are equivalent without actually having to run their computational models.

We extend the work presented in [23], where the notion of regular expressions with memory (REM) was introduced. These regular expressions can store some terminal symbols, in order to compare them later (in a string) against a new input element for (in)equality. The corresponding automata compiled from REM need to be equipped with registers. Each transition has the option to write the symbol that triggered it to some register. Transitions can also access registers to retrieve their contents (previously stored elements) and

compare them with the last element read by the automaton's head. One important limitation of REM with respect to CER is that they can handle only (in)equality relations. In this section, we extend REM so as to endow them with the capacity to use relations from "arbitrary" structures. We call these extended REM Symbolic Regular Expressions with Memory and Output (SREMO).

We assume that each terminal expression of a SREMO is a Boolean expression whose predicates are in the form of a relation P . We also assume that all possible input events constitute a universe U . We can then extend the terminology of classical regular expressions to define characters, strings and languages. Elements of U are called characters and finite sequences of characters are called strings. A set of strings L constructed from elements of U , i.e., $L \subseteq U^*$, is called a language over U . Then, a stream S is an infinite sequence $S = t_1, t_2, \dots$ where each $t_i \in U$ is a character. By $S_{1..k}$ we denote the sub-string of S composed of the first k elements of S . $S_{m..k}$ denotes the slice of S starting from the m -th and ending at the k -th element. We can then define n -ary relations P on the elements of U and use these relations, or combinations of them via Boolean connectives, as terminal expressions within a regular expression. The arguments of P refer either to the most recently read element of a string or to preceding elements, assumed to have been stored in registers. We call such terminal expressions "conditions". Conditions are the basic building blocks of SREMO. In the simplest case, they are applied to single events and act as filters. In the general case, we need them to be applied to multiple events, some of which may be stored to registers. Conditions will essentially be the n -ary guards on the transitions of SRT.

We can now define SREMO, by combining conditions via the standard regular operators. Conditions act as terminal expressions, i.e., the base case upon which we construct more complex expressions. Each condition may be accompanied by a register variable, indicating that an event satisfying the condition must be written to that register. It may also be accompanied by an output, either \bullet , indicating that the event must be marked as being part of the complex event, or \emptyset , indicating that the event is irrelevant and should be excluded from any detected complex events.

A SREMO is inductively defined as follows:

- If ϕ is a condition and o an output, then $\phi \uparrow o$ is a SREMO.
- If ϕ is a condition, o an output and r_i a register variable, then $\phi \uparrow o \downarrow r_i$ is a SREMO.
- If e_1 and e_2 are SREMO, then $e_1 + e_2$ is also a SREMO.
- If e_1 and e_2 are SREMO, then $e_1; e_2$ is also a SREMO.
- If e is a SREMO, then e^* is also a SREMO.

In order to capture SREMO, we propose Symbolic Register Transducers (SRT), an automaton model equipped with memory, logical conditions on its transitions and a single output on every transition. The basic idea is the following. We add a set of registers R to an automaton in order to be able to store events from the stream that will be used later in n -ary formulas. Each register can store at most one event. In order to evaluate whether to follow a transition or not, each transition is equipped with a guard, in the form of a Boolean formula. If the formula evaluates to TRUE, then the transition is followed. Since a formula might be n -ary, with $n \geq 1$, the values passed to its arguments during evaluation may be either the current event or the contents of some registers, i.e., some past events. In other words, the transition is also equipped with a register selection. Before evaluation, the automaton reads the contents of the required registers, passes them as arguments to the formula and the formula is evaluated. Additionally, if, during a run of the automaton, a transition is followed,

then the transition has the option to write the event that triggered it to some of the automaton's registers. These are called its write registers W , i.e., the registers whose contents may be changed by the transition. Finally, each transition, when followed, produces an output, either \emptyset , denoting that the event is not part of the match for the pattern that the SRT tries to capture, or \bullet , denoting that the event is part of the match.

2.2.2 Experimental results

We have implemented a SRT-based CER engine by extending Wayeb⁵. We present our implementation and experimental results. We present experimental results by comparing Wayeb against other state-of-the-art CER systems. Our goal is to test the systems with expressive, relational patterns, i.e., with patterns which can relate multiple events. For this reason, we had to exclude systems that cannot express relational patterns, such as CORE and previous versions of Wayeb. For some other systems, there is no publicly available implementation or the implementation is no longer maintained (e.g., CRS and Cayuga). Yet some other systems (e.g., TESLA) suffer from low performance for certain classes of queries [24].

Our comparison thus includes SASE v1.0 [25], Esper v8.7.0 [26] and FlinkCEP v1.16.1 [27]. All these engines are written in Java. Wayeb is implemented in Scala 2.12.10. All experiments were run on a 64-bit Linux machine with AMD EPYC 7543 \times 126 processors and 400 GB of memory. We used Java 1.8 for all systems. All experiments for all systems were run as single-core applications.

As a basis for our experiments, we used the benchmark suite presented in [24]⁶. The suite contains three datasets: a) stock market data from a single day (224,473 input events); b) plug measurements from smart homes (1,000,000 input events) and c) taxi trips from the city of New York (585,762 input events). For the stock market dataset, each input event is a BUY or SELL event, containing the name of the company, the price of the stock, the volume of the transaction and its timestamp. For the smart homes dataset, each input event is a LOAD event, containing a load value in Watts, a household id, a plug id and a timestamp. For the taxis dataset, each input event is a TRIP event, containing the datetime of the pickup and dropoff, the zone of the pickup and dropoff, the trip distance and duration, the fare amount, the tip amount, etc.

Since windows are ubiquitous in CER (for performance issues), we decided to focus on windowed SREMO in our experiments. We also fixed the selection strategy to skip-till-any, since this is the most demanding strategy, both in terms of time and space complexity. For all experiments described here, we have made sure that all engines produce the same results for each pattern.

The benchmark suite runs each experiment, i.e., each combination of engine, pattern and window size, 3 times. We report the average throughput and memory footprint. Throughput is measured in terms of (input) events processed per second, whereas memory is measured in terms of used memory (MB). For each run, multiple memory measurements are taken, one every 10.000 input events. Before the measurement, the garbage collector is explicitly called. We report the average of those memory measurements. The time we use to calculate

⁵ <https://github.com/EIAlev/Wayeb>

⁶ <https://github.com/CORE-cer/CORE-experiments>

throughput includes both the time required to process input events (update the state(s) of the automaton, create new runs, discard old ones, etc.) and the time required to report any complex events. However, we have slightly modified the notion of “reporting a complex event”. Instead of writing it in a file/database (a system-dependent, expensive operation), we perform (for all systems) a simple arithmetic operation on the timestamps of its constituent simple events.

Our first set of experiments is focused on sequential patterns. We begin with patterns of the following form:

$$seq_3 := ((\varphi_1(\sim) \uparrow \cdot \downarrow r_1); (\varphi_2(\sim) \uparrow \cdot); (\varphi_3(\sim, r_1) \uparrow \cdot))^{[1..w]}$$

where w is the window size and φ_i all contain “local” constraints, i.e., conditions applied to the single, most recently read event, while φ_3 relates the most recently read input event with the event that triggered φ_1 . For each such pattern, we run experiments for variable pattern “length”. We say that the length of the Pattern seq_3 is 3 because it is composed of 3 terminal sub-expressions. We can increase the length of the pattern by adding more such expressions. In our experiments we have used patterns of length 3, 6, 9 and 12.

Figure 6 presents throughput results for the aforementioned sequential patterns and for all datasets. Wayeb and Esper stand out clearly as the most efficient engines in terms of throughput. Wayeb also has a significant advantage over Esper in most experiments and a slight advantage for the smart homes dataset. For example, Wayeb is almost 2.5 times as efficient as Esper for the taxis dataset. Wayeb has a slightly better performance than Esper, its main competitor in terms of throughput. In general, we see that the performance is relatively stable as a function of pattern length for all systems.

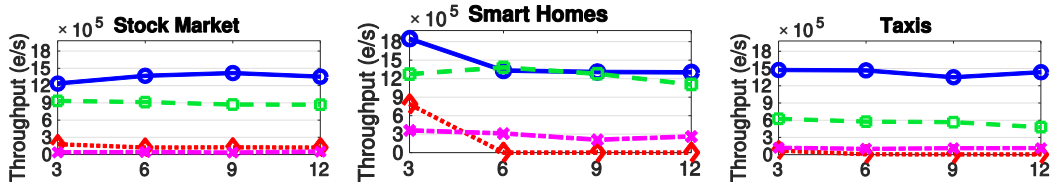


Figure 6 Throughput for sequential patterns with n-ary predicates as a function of pattern length. Window sizes are $w_{stock} = 500$, $w_{smart} = 5$, $w_{taxi} = 100$

In the next set of experiments, we investigated the behaviour of all systems for increasing window sizes. For each dataset, we increased the window size up to the point where throughput exhibits a significant drop. **Figure 7** shows the relevant results. Wayeb again exhibits the best performance in terms of throughput, followed by Esper. All systems exhibit a throughput deterioration as the window size increases. This implies that the window size is more factor in determining the number of created runs than pattern length.

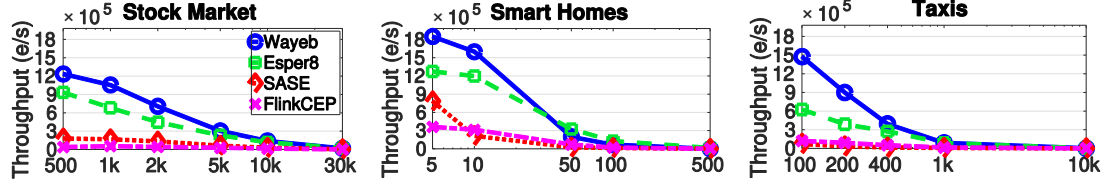


Figure 7 Throughput for sequential patterns with n-ary predicates as a function of window size. Pattern length is 3

In the last set of experiments, we used the stock market dataset and tested all engines against patterns with various operators. We considered a diverse range of patterns, where other operators like disjunction, iteration and their combination were employed. In particular, we tested 5 patterns: q_1) A sequential pattern starting and ending with a SELL event, and with two BUY events in between; q_2) same as q_1 , but with local thresholds on price; q_3) same as q_1 , but now includes disjunction; q_4) same as q_3 , but with local thresholds on price; q_5) combining iteration and disjunction.

SASE can only support SREMO q_1 and q_2 . Therefore, we do not show SASE results for SREMO q_3 , q_4 and q_5 . FlinkCEP supports all 5 patterns, but its semantics of the iteration operator are ambiguous and its results when using iteration do not match those of the other systems. Therefore, we do not show FlinkCEP results for SREMO q_5 .

The relevant results are shown in **Figure 8**. Wayeb has the highest throughput for all patterns, followed by Esper. The performance for q_2 is higher than that for q_1 , due to the presence of extra threshold filters which prune several runs. On the other hand, q_3 is the most demanding one, because it does not have any threshold filters and it includes disjunction, thus leading to more runs being created. q_4 rebounds to higher throughput figures, due to the inclusion of filters. For q_5 , Esper has its lowest performance and Wayeb its second lowest, due to the presence of both iteration and disjunction.

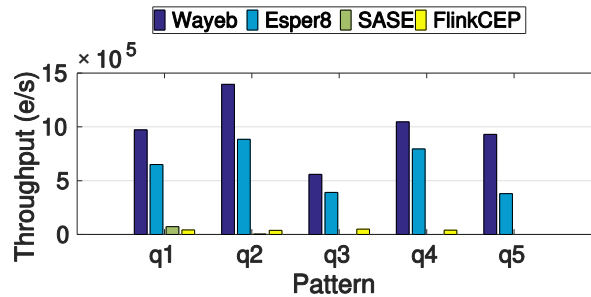


Figure 8 Throughput for patterns with n-ary predicates and various operators.

$w = 1000$

2.3 Complex Event Recognition with Allen Relations

Contemporary CER applications require the processing of large, high-velocity streams of symbolic events derived from sensor data, in order to detect and report the satisfaction of complex event patterns with minimal latency. In maritime situational awareness, e.g., a CER system consumes streams of vessel position signals, in order to detect instances of dangerous, suspicious and illegal vessel activities in real time, thus supporting safe shipping [28]. The target activities of a CER system, such as illegal fishing, are typically durative, and thus should be expressed using temporal intervals. Moreover, the use of Allen's Interval Algebra has proven quintessential for CER [29] [30]. Allen's algebra specifies thirteen jointly exhaustive and pairwise disjoint relations among intervals [31]. Consider, e.g., the detection of the vessel activity 'disappeared in area', where a vessel may be attempting to conceal illegal activities in a certain area, such as fishing in fisheries restricted areas, by stopping transmitting its position. This phenomenon can be expressed with the 'meets' relation of Allen's algebra, while it cannot be captured by common interval operators, such as union and intersection.

The Event Calculus is a logic programming formalism for representing and reasoning about events and their effects over time [32]. The Event Calculus exhibits a formal, declarative semantics, while supporting non-monotonic reasoning with background knowledge, relational events and hierarchical event patterns. These types of temporal specifications are commonly required in CER [10]. The "Run-Time Event Calculus" (RTEC) is a formal, logic-based computational framework for online CER [33]. RTEC includes optimisation techniques, like windowing, allowing for highly efficient reasoning in CER applications. We proposed RTEC_A, an extension of RTEC that supports the relations of Allen's interval algebra in complex event patterns.

In order to express complex event patterns with Allen relations, RTEC_A supports logic programming rules with head $holdsFor(F = V, I)$, expressing that a fluent F has value V in the maximal intervals of list I . We use a fluent-value pair (FVP), such as $F = V$, to denote a complex event. In order to incorporate Allen relations, the rules with head $holdsFor(F = V, I)$ may contain body predicates in the form of $allen(rel, S, T, outMode, I)$, where rel denotes an Allen relation, S and T are input lists of maximal intervals, $outMode$ expresses how we should treat the interval pairs (i_s, i_t) satisfying rel , where $i_s \in S$ and $i_t \in T$, and I is the output list of maximal intervals. Consider the following example rule:

$$\begin{aligned} holdsFor(disappearedInArea(Vl, AT) = true, I_{dia}) \leftarrow \\ holdsFor(withinArea(Vl, AT) = true, S), \\ holdsFor(gap(Vl) = farFromPorts, T), \\ allen(meets, S, T, target, I_{dia}). \end{aligned}$$

$disappearedInArea(Vl, AT) = true$ is an FVP expressing the intervals I_{dia} during which vessel Vl stopped transmitting its position while in an area of type AT . The first two conditions of the above rule express that Vl is within an area of type AT in the intervals of list S and that Vl has stopped transmitting its position while being in the open sea in the intervals of list T . The last condition of the rule expresses the meets Allen relation. $allen(meets, S, T, target, I_{dia})$ states that from the interval pairs (i_s, i_t) satisfying meets,

where $i_s \in S$ and $i_t \in T$, we will keep in the output list I_{dia} the “target” intervals, i.e., the intervals of the second input list T . Therefore, a vessel VI is said to disappear in an area of type AT during an interval i_{dia} , if i_{dia} is an interval during which $gap(VI) = farFromPorts$, i.e., VI stopped transmitting its position while being in the open sea, and i_{dia} is met by an interval during which VI was within an area of type AT .

We describe $RTEC_A$ in [34], where we outline the syntax, semantics and reasoning algorithms of $RTEC_A$, demonstrating their correctness and linear-time complexity. Moreover, we present an extensive, reproducible empirical comparison of our approach with two state-of-the-art systems supporting Allen relations on real maritime data. Our comparison demonstrates that $RTEC_A$ is at least one order of magnitude more efficient than the state of the art.

2.4 Components for Critical Maritime Event Forecasting and Resolution

Two distinct components for forecasting and resolving critical complex maritime events are being developed by Kpler for the Maritime Use Case:

1. MAR_1: Collision forecasting and rerouting
2. MAR_2: Hazardous weather routing

2.4.1 MAR_1: Collision forecasting and rerouting

In 2020 alone 2632 accidents occurred in European Waters according to European Maritime Safety Agency. In CREXDATA, Kpler aims to increase maritime safety by developing components and tools for the early forecasting of critical maritime events and for the automation of the decision-making process for resolving critical situations at sea. Kpler has already developed an approach for vessel collision forecasting that is integrated with the distributed system architecture based on the Akka processing engine [35]. In CREXDATA, in the context of the Maritime Use Case, Kpler develops a solution for vessel collision avoidance for manned and unmanned vessels.

The vessel collision avoidance solution is based on the Frenet Frame Optimal Trajectory Generation algorithm. The algorithm is advantageous for solving dynamic routing problems in motion planning in complex environments, for robotics, autonomous vehicles and maritime navigation.

The Frenet Frame Optimal Trajectory Generation algorithm addresses the dynamic routing problem by offering an optimal control-based solution for motion planning. It features a reactive collision avoidance algorithm suited for complex dynamic environments and manages long-term objectives such as velocity keeping, route keeping, and stopping, while considering vessel-specific parameters [36]. Additionally, the COLREGs (International Regulations for Preventing Collisions at Sea) are integrated with the collision avoidance algorithm, so that the algorithm produces COLREG-compliant path planning solutions. Specifically, the integration of COLREG involves modeling and incorporating vessel safety zones, detecting vessel-to-vessel interaction cases, and filtering out non-COLREG compliant trajectories during the trajectory generation process (**Figure 9**).

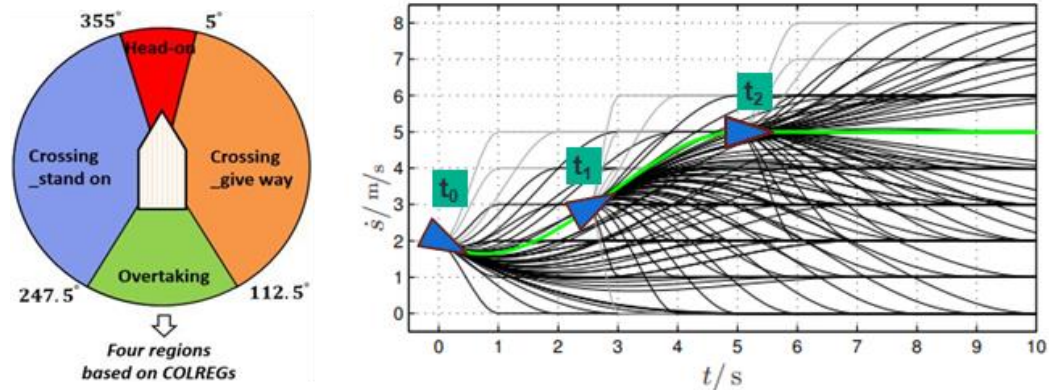


Figure 9: Left: COLREG regions for vessel-to-vessel interaction classification [37]. Right: Optimal path planning and velocity adaption of a vessel with green being the final COLREG compliant optimal trajectory generated in each subsequent replanning step, black the COLREG compliant valid trajectories, and gray the invalid alternatives in each replanning step t . Image adapted from [36]

Figure 10 presents the workflow of the vessel collision avoidance algorithm. **Figure 11** presents an example output of the collision avoidance algorithm given two vessels that are forecasted to collide head on at the detected collision position. First, the collision detection data [35] is ingested from the Redis database, the CREXDATA platform (simulated collision events for the simulator tools and the interactive simulation scenarios) and the autonomous vessel (sea trial). The input includes the information on the collision detection event and vessel specific dynamic and static information.

Subsequently, the type of COLREG interaction is identified according to the interacting vessel speeds and courses. Finally, the compliant trajectories are identified by the frenet path planning algorithm.

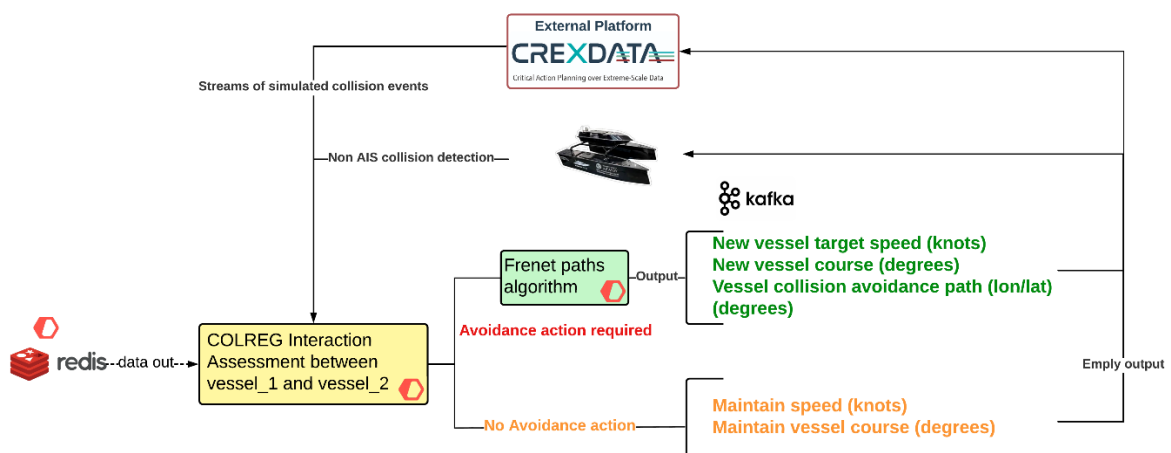


Figure 10: Vessel collision avoidance workflow

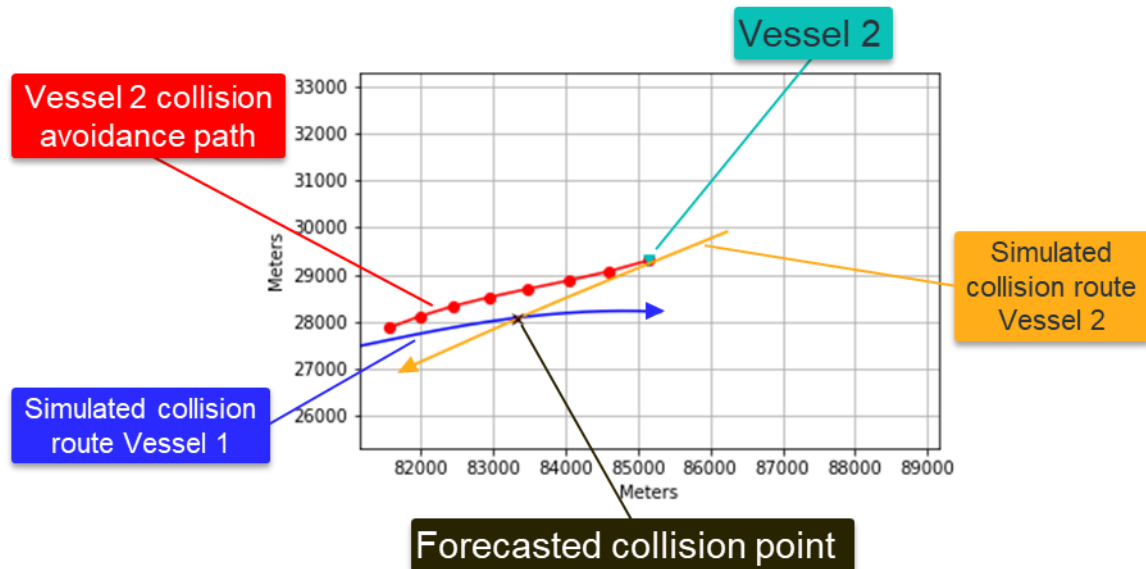


Figure 11: Example of a head on collision and the generated COLREG compliant collision avoidance path

2.4.2 MAR_2: Hazardous weather routing

Collective vessel fleet intelligence involves using historical weather data and weather forecasts with historical mobility information to address routing challenges in critical sea weather events. In the context of the Maritime Use Case, Kpler develops a hazardous weather routing solution for manned and unmanned vessels. The hazardous weather routing solution for vessels sailing around the world holds immense significance, particularly in the context of international vessel traffic, safety, and the minimization of route disruptions that impact the global supply chain. Extreme weather events, which are becoming increasingly frequent and severe due to climate change, pose significant risks to maritime operations. Maritime traffic routes are vital for the trade of oil, liquefied natural gas (LNG), and other essential commodities. For example, ports along the Gulf of Mexico and the US East Coast, such as Corpus Christi, Houston, and Beaumont, are crucial hubs for the export of these resources to European markets. Disruptions along these routes due to hazardous weather conditions can have far-reaching consequences (see: **Figure 12** and **Figure 13**).

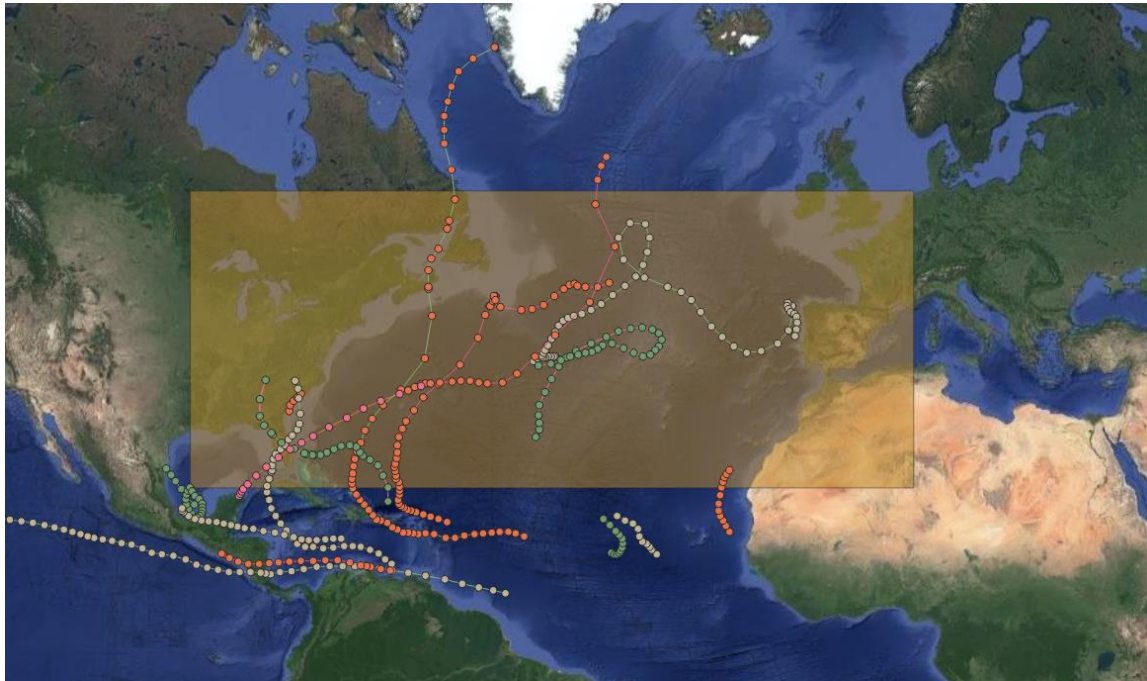


Figure 12: Visualization of the 2022 hurricane events (their paths) along the North Atlantic Ocean

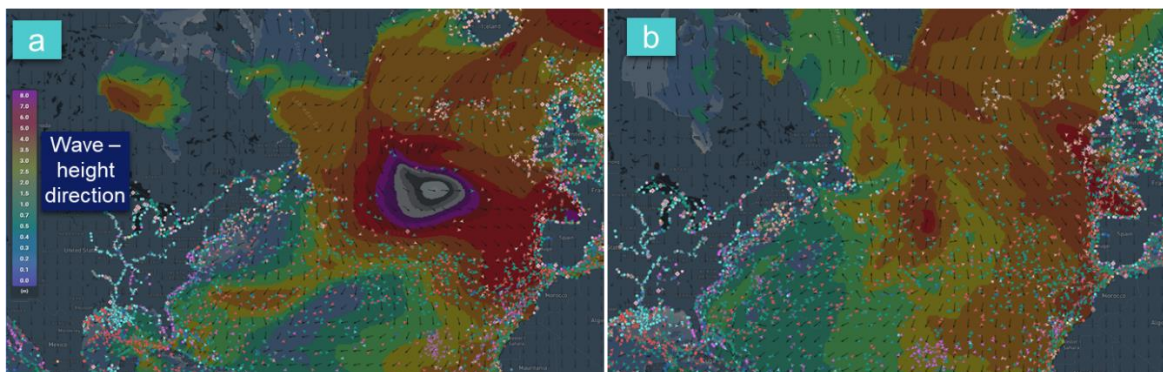


Figure 13: a) Disruption of maritime traffic caused by Storm Ciarán that severely affected parts of Europe and the North Atlantic from late October to early November 2023. b) Comparison with the 6.11.2023, after the storm has calmed under normal weather conditions. Heatmap visualizes the wave height – direction. Different vessel types sailing in the area are visualized as triangles with colours indicating the different vessel type

The weather routing approach entails modeling the sea space between origin-destination locations using the H3 hexagonal geospatial index (h3geo.org) [38]. The H3 index resolves the sea area into a grid. Each H3 cell (hex size: 5) of the index forms a node and is interconnected with edges to each neighboring cell. This forms a high-resolution graph with nodes and edges with uniform index size that fully covers the entire globe, and which can be used as the solution space for the hazardous weather routing algorithm.

Subsequently an ETL process has been built in Airflow for the weather data and forecast extraction, fusion with the H3 grid and scaling transformation. The ETL process extracts weather data from NOAA (wind data) and from Copernicus (wave and currents data), standardizes it into a uniform format and fuses it with the H3 grid.

The resolution of the weather features is lower than the resolution of the H3 grid. Thus, it is important to map the respective weather features accurately across the entire H3 grid fill the missing areas accordingly. The mapping of the weather features on the H3 index is facilitated with following steps and is visualized in **Figure 14**:

1. Weather features are mapped directly into their corresponding H3 cells. (purple cells: 1st level)
2. 2nd level neighbouring (green) cells are assigned with the value from the 1st level cells
3. 3rd level red cells are assigned the average of the respective H3 2nd level neighbours
4. 4th level yellow cells are assigned with the neighbouring average values.

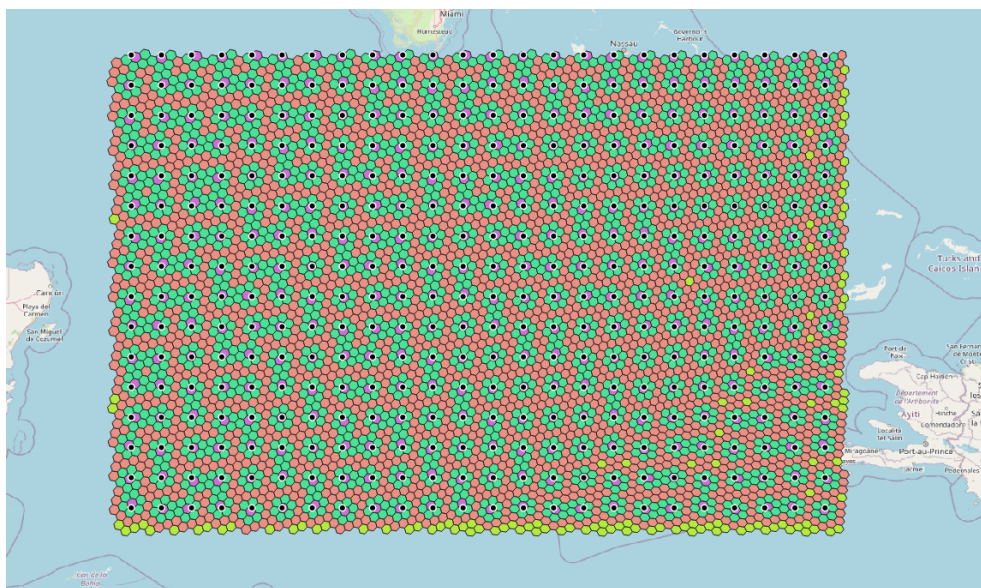


Figure 14: Mapping of weather features on the H3 index (example for NOAA wind dataset)

It is important to note that the three weather datasets for wind, waves and currents have different recording intervals. NOAA provides wind data every six hours with hourly datasets, Copernicus supplies wave data twice a day (dividing the day in half) with three-hourly datasets and offers currents data once a day with hourly datasets. The data extraction is scheduled to run daily, executing various batch jobs that automate and periodically perform the specified tasks and updates to the current weather databases used for hazardous weather routing (see **Figure 15**).

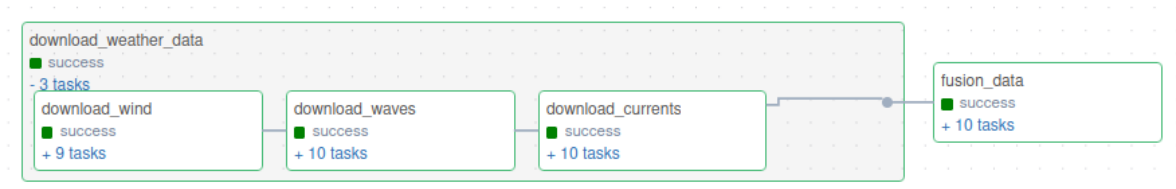


Figure 15: Download and fusion process of weather data from NOAA and Copernicus (Task groups)

Additionally, a statistical data extraction of historical weather data is performed on data from September 2022. Statistics for wind and currents are generated by calculating the magnitudes for the wind and currents, according to the respective official scales for categorizing the intensity of weather features. Subsequently, the weather data is scaled across the entire area by utilizing the minimum and maximum values from the statistical extraction (see **Figure 16**).

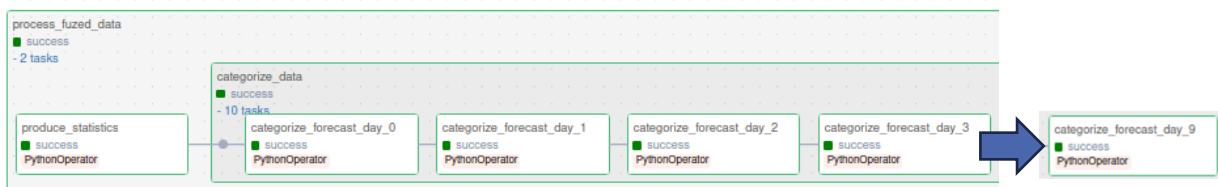


Figure 16: Statistics extraction and weather forecasts scale categorization process step for each weather parameter. Example showcases the task groups for days 0 (current) to 3 (Total task groups reach up to day 9)

The solution leverages historical AIS data fused with the historical weather data and weather forecasts. Based on the statistical weather analysis and scaling, weights along the edges across the different H3 nodes are generated. These weights are treated as functions of the forecasted changing weather conditions, leading to the generation of updated and safe vessel routes toward the destination. The final implemented routing solution is graph-based and employs the A* algorithm for the route definition. The implemented approach for the hazardous weather routing solution is presented in **Figure 17**.

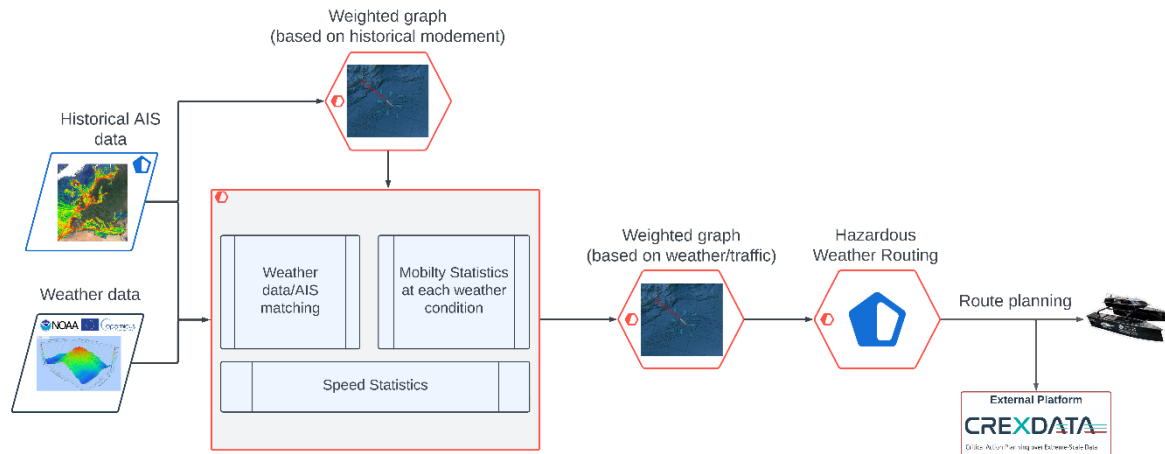


Figure 17: Hazardous weather routing methodological workflow

Initial evaluation results for the hazardous weather routing component are presented in Chapter 5 Maritime Use Case in Deliverable 2.2.

3 Interactive Learning for Simulation Exploration

In this section we present our work in the scope of “T4.2 Interactive learning for simulations exploration”. The main objectives of T4.2 are:

- Develop algorithms for guiding large-scale simulations towards desired ends.
- Simulate course corrections and adaptations to a changing environment.
- Avoid exhaustive search of the solution space by incorporating faster and effective solutions.
- Promote interactive learning to allow users to be more active in guiding the simulations.

In the first half of the project, we focused on defining the interactive learning and simulations exploration scenarios for each use-case of CREXDATA. Having the simulators in place (see D2.2), we examined what would be relevant and useful for the end-users, given the interactive learning possibilities that each simulator and use-case definition provides.

In the following, for each use-case, we briefly describe the simulators used and focus on the T4.2's perspective regarding the parameters that are calibrated and explored. Also, we define several scenarios and outline the methods that are going to be used for simulations exploration, as well as for aiding the end-users in an interactive learning manner.

3.1 Emergency Use-Case

In the first half of the CREXDATA project, a focus was set in the weather induced Emergency Case (EmCase) to “weather related simulation” (cf. [D2.1, p.28]). MIKE+ was selected as a simulator engine (see details in D2.2). The software is available with a research license, offering an API to invoke simulation runs and access output results. It is a representative sample of similar simulators, also preparing for different natural phenomena like forest fires. Specific use cases cover the three use case types of prediction, calibration and optimization.

The conceptual software architecture (see **Figure 18**) for the integration of MIKE+ covers the KAFKA-based interface between the CREXDATA system including the T4.2 component, ARGOS as a kind of mediator system invoking simulations and pushing simulation results, as well as a sample deployment of MIKE+ extended by a batch process. This batch process starts a simulation run, uniquely identified by a simID, converts output results from DFS formats to T4.2 format, and merges all relevant result packages. The software packages [MIKE+ Py](#) and [MIKE IO](#) are used for this purpose.

MIKE+⁷ manages projects through so called project (MUPP) Files, loading context data and parameter values from an SQLite database. From that database, further data sets like Digital Elevation Models (DEMs) are acquired (in that case, using SpatiaLite). Simulations are run

⁷ URL https://manuals.mikepoweredbydhi.help/latest/Cities/MIKE_Plus_Model_Manager.pdf for general and esp. network modelling including results specifications, URL https://manuals.mikepoweredbydhi.help/latest/Cities/MIKE_Plus_2DOverland.pdf for 2D overland modelling including results specification, last access 20.05.2024

based on data for three types of parameters: context data (see **Figure 19**), configuration settings and input data.

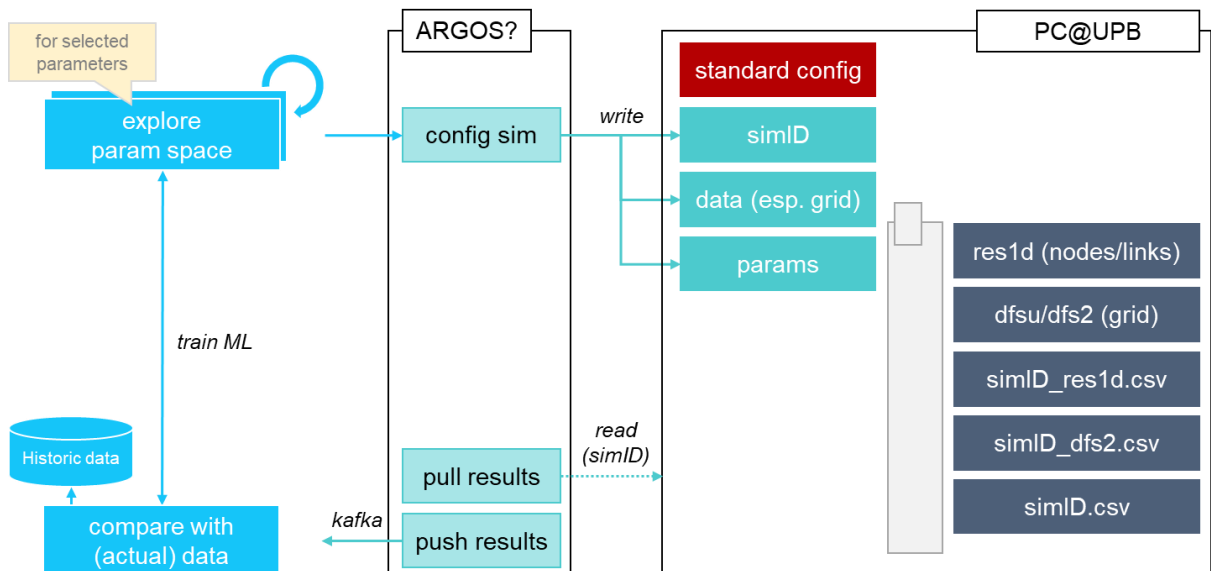


Figure 18: Adoption of the simulator architecture [D2.1]

3.1.1 Input parameters and data

Simulations are conducted based on context data which mainly subsume

- 2D overland (grid/raster data):
 - Geographical information setting references to world coordinates (e.g., by lat-lon coordinates of the south-western edge).
 - Bathymetry resp. Digital Terrain Model (DTM) which indicates the surface level (in case of rivers, lakes and oceans below water).
 - Digital Surface Model (DSM) indicating the surface that includes buildings etc.
- Catchments and/or zones, breaking down the grid into areas with similar properties:
 - Buildings
 - Streets
 - Lawns etc. (e.g., by layer of land cover)
- Collection network system (sewage system)
 - nodes (manholes, outlets and basins): defined by coordinates, type, diameter, ground and bottom level etc.; specified by, e.g., max. and min. infiltration rates
 - circular pipes, defined by references to two nodes, type, height, width, diameter, length etc.

Parameters in configuration settings are assumed to be stable in general for simplification purposes. The latter means that, in CREXDATA use cases, they will not be changed in between simulation runs by decision. Exemplary configuration parameters are:

- General parameters:
 - Temporal resolution for the simulation of output parameters like water level, flow velocity etc.
 - Spatial resolution of the grid (e.g., grid elements in 5 meters distance)
 - Catchments
- Parameters stable by decision, esp. properties of catchments and/or zones determining, e.g., the rainfall-runoff model, which is based on temperatures and drought conditions of the ground (determined by meteorology, i.e., potentially calculated by impact assessments based on meteorological data)
 - Dampening delta depth
 - Weir coefficient
 - Initial values for the hydrodynamic variables on dryness resp. water level
 - Surface roughness, eddy viscosity etc.

Main input data for simulations might result from sensing, especially acquired by satellites (grid/raster data) and weather stations (time series data), and from forecasting (typically grid/raster data), gathered from forecasting services running models of single- or multi-phenomena models. Such input data refers to

- Meteorological input data parameters – precipitation (type: rainfall and raingauge): main influence for flooding events, determined by intensity over time
- Meteorological input data parameters – wind: direction and intensity

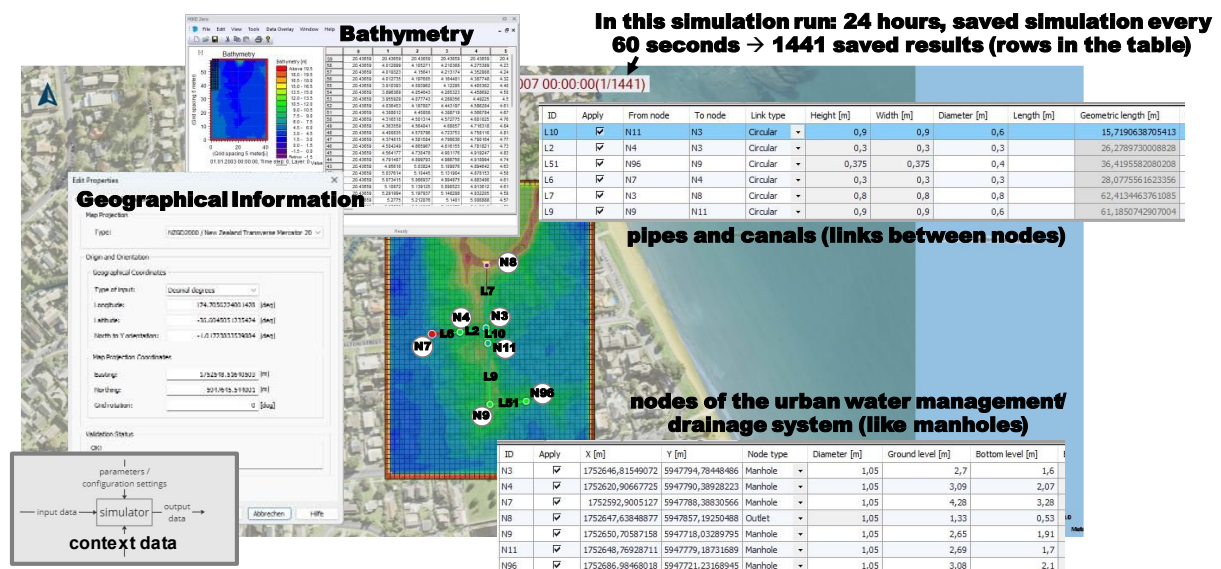


Figure 19: Visualisation of an excerpt from the context data used for simulations

3.1.2 Output parameters

The simulation results in output data, carrying time series for specific attributes for the grid and for networks⁸:

- Network: nodes and pipes
 - focused in CREXDATA: total water depth (water level)
 - further parameters like discharge to surface and (optional) volume
- Network: links
 - further parameters like flow velocity
- Network: nodes
 - further parameters like external water level and diverted runoff to surface
- Elements (cells) of the 2D overland grid/raster (incl. catchments)
 - focused in CREXDATA: water level
 - further parameters like volume balance and discharge (through cross-section), U/V velocity and current speed

	A	B	C	D	J	K	L	Q	R	S	T	U	V	W
1														
2	time stamp	item name / EUM quantity	N11	N3	nodes ...	L10	L2	links ...	grid element ->					
1443	03:54:00	Total water depth (meter)		2.403.383		24.031.506	2.403.383							
1444	03:55:00	Total water depth (meter)		2.403.866		24.037.097	2.403.866							
1445	03:56:00	Total water depth (meter)		24.043.808		2.404.288	24.043.808							
1446	03:57:00	Total water depth (meter)		24.046.068		24.044.785	24.046.068							
1447	03:58:00	Total water depth (meter)		24.049.394		24.048.226	24.049.394							
1448	03:59:00	Total water depth (meter)		24.054.675		24.054.432	24.054.675							
1449	04:00:00	Total water depth (meter)		24.059.458		24.059.722	24.059.458							
1450	00:00:00	Water level							32.815	20.715	1.604 0.534		1.913	1.701
1451	00:01:00	Water level							32.815	20.715.988	1.604 0.534		19.131.045	1.701
1452	00:02:00	Water level							32.815	20.716.789	1.604 0.534		1.913.222	1.701
1453	00:03:00	Water level							32.815	2.071.743	1.604 0.534		1.913.338	1.701
1454	00:04:00	Water level							32.815	20.717.947	1.604 0.534		19.134.544	17.030.081
1455	00:05:00	Water level							32.815	20.718.453	16.059.788 0.53430986		19.138.448	17.030.654
1456	00:06:00	Water level							32.815	20.719.256	16.164.229 0.53754437		1.916.998	17.034.534
1457	00:07:00	Water level							32.815	20.720.203	16.308.837 0.60867727		19.289.764	17.063.141

Figure 20: Output data from a sample simulation run (executed for a duration of 4 hours, in equal time steps of 60 seconds each) for total water depth in nodes (N11, N3, ...) and links (L10, L2, ...) in time steps 00:3:54:00 to 04:00:00 from res1d file and water level for grid elements in time steps 00:00:00 to 00:07:00 from dfsu/dfs2 file

Relevant data for interactive learning scenarios (T4.2) and, potentially resulting from that, simulation for visualization in Augmented Reality (T5.4) subsumes both parameters in configuration settings and input data parameters. In general, both weather data from sensors as well as weather forecasts is available through ARGOS, acting as a kind of proxy to services like Copernicus, ECMWF, Meteostat etc.

3.1.3 Use cases in the EmCase

Real data is available for the city of Innsbruck, made available through an agreement between the CREXDATA consortium (DCNA, UPB, NCSR und TUC) and Innsbrucker Kanalbetriebe (IKB). IKB provided the entire context data including the sewer system and catchments as well as precipitation and measurement data for an extreme weather even in

⁸ There are further options, for instance, for nodes and links (like water quality) or junctions and tanks (pressure, head and water demand) which are not reflected here.

2016 (for details see D2.2). The city of Innsbruck does not run a simulation model like MIKE+, so that high efforts would be required to create the initial model.

To avoid these efforts in the first stage, sample conditions are assumed for the initial phase of CREXDATA⁹. Offline calibration would be required, in actual use, for parameters in configuration settings that are determined as “stable by decision” above.

Initial Interactive Learning scenarios are presented in **Table 2**, encapsulated in use cases for T4.2. Each of these use cases refers to a specific type of Parameter Exploration (interventions). Relevant output describes attributes that are help in action planning and decision making, indicating most promising measures resp. most critical situations. For these attributes, historic data might be relevant (cf. **Figure 18**) as reference data (esp. for end users interacting with a T4.2 service) or training/test data (for machine learning models). We note that from the list of the possible scenarios, in the next half of the project we will focus on particular ones based on various criteria, such as the required preparatory effort, opportunities for further integration in other tasks, etc.

Table 2: Overview of possible interactive learning scenarios for the EmCase.

No.	<u>Interactive Learning scenarios (use cases)</u>	<u>Parameter Exploration (interventions): modify configuration or input parameters</u>	<u>Relevant output</u>
1	Having an unexpected actual total water depth in a node or in a link, or a water level in a grid element, does that indicate that the sewer network does not work as expected?	Delete node (covered manhole), link (blocked pipe) or disable discharge through cross-section (between grid elements/catchments)	Actual water level measurements (→ find blockage)
2	Where should barriers be set to reduce critical water levels within the grid or water depth within network elements?	Elevate grid elements in the DSM to indicate barriers (e.g., by loading pre-defined alternative grids, each with a prepared response measure)	Find minimal average water level or total water volume in the entire grid
3	Expecting an extreme weather event, which of the prepared risk scenarios is valid and, as a consequence, which management plan shall be activated by a decision?	Vary precipitation over time ← options: default scenarios, historic scenarios from ARGOS, alternative forecasts from ARGOS	Indicate expected probability level (e.g., 100y flooding)

⁹ Initially given by a sample project provided by DHI for MIKE+.

4	Is it necessary to consider wind situations when assessing impacts?	Vary wind speed and direction ← ARGOS	Indicate most problematic wind conditions
5	Does the forecast match the actual conditions in terms of water discharging to the ground or evaporating to the air?	Specify possible discharge/evaporation (?) relative to temperature/ drought ← ARGOS	Confirm simulations with regard to actual/measured drought
6.1	Rescuer team needs to move from starting point to a point in the emergency scene and must decide on a safe entry route	Vary precipitation over time (cf. use case 3)	Routes ¹⁰ that are not (or least likely) flooded in any simulation scenario
6.2	Rescuer team needs to move from a point in the emergency scene to a safe place and must decide on a safe rescue route ¹¹	Vary precipitation over time (cf. use case 3)	Routes that are not (or least likely) flooded in any simulation scenario

3.2 Health Crisis Use-Case

The second CREXDATA use-case is divided into two sub-use-cases, one focusing on multiscale simulations for the condition of the lungs of COVID19 patients, and the second regards an epidemiological scenario that takes into account mobility data and historical measurements relevant to the COVID19 pandemic.

3.2.1 Multiscale Simulations

This use case is based on Alya and PhysiBoSS to simulate COVID-19 dynamics in the lungs and airflow in the upper airways (see details in D2.1). It aims to understand varying COVID-19 severity and optimize interventions and treatments using CREXDATA technologies. The process involves creating a 3D model of the upper airways, simulating infection dynamics, and integrating omics data into the model. The integration and analysis phase is crucial in this project. The simulations from Alya and PhysiBoSS will be combined to analyse the progression of COVID-19, providing insights into disease spread and progression in the lungs. These insights could inform new treatment strategies or preventive measures. Complex Event Forecasting is being used to predict complex events based on historical data, enhancing our understanding of disease progression. Furthermore, Interactive Learning for Simulation Exploration will allow users to interactively explore and learn from simulations, fostering a deeper understanding of the disease dynamics and potentially leading to breakthroughs in treatment approaches. The overview of our approach is shown in **Figure 21**. This phase embodies the synergy of simulation and AI in combating COVID-19.

¹⁰ requires routing service, as well as mapping of route and “road” elements in the project

¹¹ similar to suggesting a safe exit route to citizens, e.g., during an emergency call

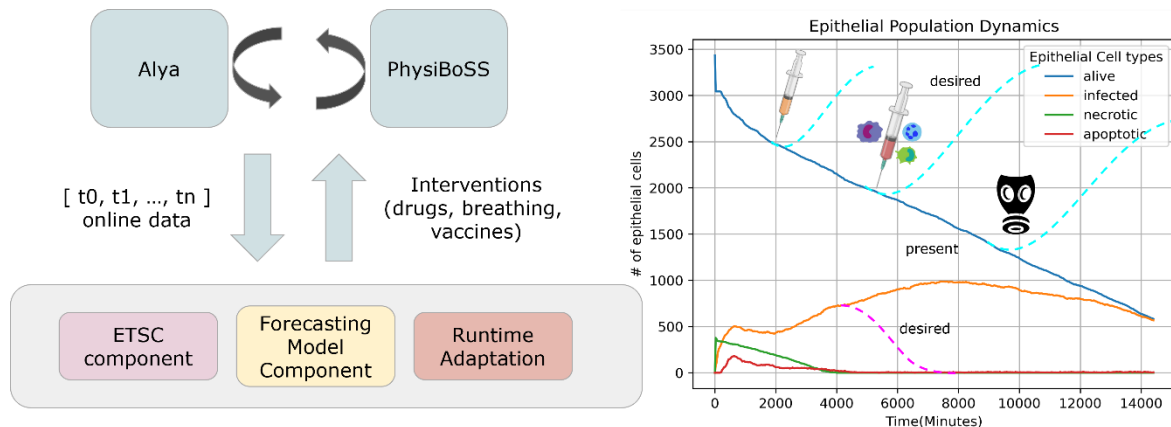


Figure 21: (Left) A schematic of the interconnection between PhysiBoSS and Alya, the simulators, with the different components from the WP4 that will help with the calibration, the exploration and the interventions. (Right) Time-series example and illustration of the alternative interventions

The simulators that will be used in this task are PhysiBoSS and Alya. PhysiBoSS is a multi-scale agent-based modelling framework that integrates physical dimension and cell signalling [39, 40]. It provides a flexible and computationally efficient framework to explore the effect of environmental and genetic alterations of individual cells at the population level. PhysiBoSS is designed to bridge the critical gap from single-cell genotype to single-cell phenotype and emergent multicellular behaviour. It's particularly useful when studying heterogeneous population response to treatment, mutation effects, different modes of invasion or isomorphic morphogenesis events.

On the other hand, Alya is a high-performance computational mechanics code designed to solve complex coupled multi-physics/multi-scale/multi-domain problems, which are mostly coming from the engineering realm [40]. It solves a variety of physics including incompressible/compressible flows, non-linear solid mechanics, chemistry, particle transport, heat transfer, turbulence modelling, electrical propagation, and more. Alya was specially designed for massively parallel supercomputers, and the parallelization embraces four levels of the computer hierarchy. It's used in a variety of engineering simulations and is part of the Unified European Applications Benchmark Suite.

In this project, we will perform grid search for the estimation of simulator parameter values, coupled with ETSC algorithms for the timely recognition of non-relevant ones. The non-relevant simulation instances will then be prematurely terminated before their completion and thus save computational time and resources and speed-up the whole calibration process. These parameters include the initial concentration of oxygen, the number of virions arriving at the alveoli, and the state of the alveoli (represented by the percentage of healthy cells in the simulation). The calibration of these parameters is crucial as it influences the accuracy of the simulations and the subsequent analysis. By fine-tuning these parameters, we aim to create a more precise and representative model of COVID-19 progression, which in turn can lead to more effective treatment strategies.

Likewise, Interactive Learning interventions will be crucial in optimizing three key aspects of COVID-19 treatment. Firstly, we aim to determine the optimal timing for the delivery of mechanical oxygen support. This is critical in ensuring patients receive necessary support

at the most beneficial time. Secondly, we will identify the ideal time for drug delivery, which could significantly impact the effectiveness of the treatment. Lastly, we will ascertain the optimal drug concentration, balancing efficacy and potential side effects. These interventions aim to personalize and enhance COVID-19 treatment, potentially improving patient outcomes by leveraging different AI tools developed in CREXDATA. For instance, we will continue our work in analysing simulation results using online data and different optimization methods, such as Genetic Algorithms, and Covariance Matrix Adaptation [41] [42].

3.2.2 Epidemiological Scenario

For the Epidemiological Scenario, we will use the MMCAcovid19 to simulate the spatiotemporal patterns of COVID-19 progression during the pandemic in Spain. The Epidemiological Scenario is divided into two problems that aim to be solved by integrating different CREXDATA technologies. The first problem is the calibration of epidemiological and population parameters to define a reference model for simulating the spread of COVID-19 in Spain. In the second problem, we will use the calibrated model to evaluate different interventions to reduce the epidemic's impact, including i) designing confinement strategies, ranging from national-level lockdowns to region-specific restrictions, and ii) assessing the effectiveness of various vaccination strategies.

The MMCAcovid19-vac is a software package for simulating the spread of infectious diseases in a metapopulation, considering different types of agents (e.g., various age groups), their interactions, and daily mobility patterns. The simulator relies on the Microscopic Markov Chain Approach, with more details available in D2.2. The simulator will be integrated into a model exploration workflow designed to characterize large parameter spaces. **Figure 22** illustrates the different components of the workflow, including the simulator, the model exploration component, the algorithms used for parameter exploration, and the interactions between these components. Additionally, the workflow establishes a common architecture that will be used to address various scenarios. More details about the model exploration workflow can be found in D2.2.

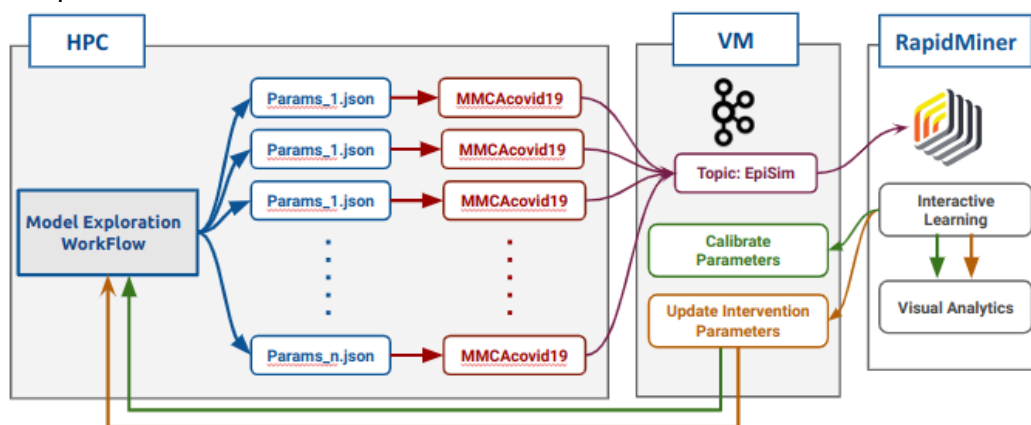


Figure 22: Model exploration workflow for the epidemiological scenario

As mentioned, this use case includes two scenarios: i) the calibration of epidemiological parameters; and ii) the design of effective interventions to control the disease spread. The calibration of parameters consists of finding parameters that when plugged into the model the simulation reproduced the observed pattern in the number of new COVID-19 cases, hospitalizations and fatalities for different age groups and different regions.

We first focused on a subset of epidemiological parameters described in **Table 3** which includes a brief description and the type of data. In this stage of the project, we have focused on calibrating a subset of epidemiological parameters rows (highlighted in grey). The calibration process values incorporate real-world data which is used to guide the parameter search. For this, we have collected a comprehensive dataset that includes COVID-19 reports that include new cases, fatalities and hospitalizations reported daily and weekly at different levels of spatial resolution (e.g. country-level, provinces, municipalities) together with phone-based anonymized daily mobility data.

Table 3: Epidemiological parameters of the MMCA Covid19-vac infectious model

Parameters	Description	Data Type
β^I	Infectivity of symptomatic ($S \rightarrow E$)	scalar (float)
β^A	Infectivity of asymptomatic ($S \rightarrow E$)	scalar (float)
η^g	Exposed (E) rate for each age group	vector float (1xg)
α^g	Asymptomatic (A) infectious rate for each age group	vector float (1xg)
μ^g	Infectious (I) rate for each age group	vector float (1xg)
θ^g	Direct death probability ($I \rightarrow PD$)	vector float (1xg)
γ^g	ICU probability ($I \rightarrow PH$)	vector float (1xg)
ζ^g	Pre-deceased (PD) rate	vector float (1xg)
λ^g	Pre-hospitalized (PH) in ICU rate	vector float (1xg)
ω^g	Fatality probability in ICU ($PH \rightarrow HD$)	vector float (1xg)
ψ^g	Death (HD) rate in ICU for each age group	vector float (1xg)
χ^g	ICU (HR) discharge rate for each age group	vector float (1xg)

To guide the calibration of parameters, we are currently using various optimization and machine learning techniques, including Genetic Algorithms (GA) and Covariance Matrix Adaptation (CMA). Additionally, we are employing approaches such as active and interactive learning combined with visual analytics to improve the characterization of the parameter space.

Specifically, we are developing a workflow that identifies candidate parameters, and the simulations produced by these parameters are analysed using visual analytic techniques. This allows expert users to identify the most relevant parameter sets and discard those that initially seemed promising but, upon deeper analysis, reveal inconsistencies when compared against real-world data. We are also evaluating the use of Federated Learning to perform Approximate Bayesian Computation in a federated manner, aiming to learn the posterior distribution of parameters in context when different users are interested in learning parameters but the data used to guide the parameter exploration is sensitive and thus, cannot be exchanged for privacy concern.

The second scenario will use the workflow to find effective interventions to control the spread of the disease in different situations. Specifically, we will focus on two types of interventions:

i) control measures for mobility reduction, social distancing, and household permeability, and
ii) the introduction of vaccination campaigns using different technologies developed in the context of the CREXDATA project. **Table 4** shows the parameters used to describe these interventions.

The group of rows at the top of **Table 4** describes the parameters used to model mobility reductions and social distancing policies. These parameters include the periods during which the policy is applied, scalar values to model social distance reductions for each age group, the permeabilities of confined households, and a time series with values for the percentage of mobility reduction. The group of rows at the bottom of **Table 4** describes the parameters used to model the introduction of vaccines, including the start date of vaccination, the duration of the campaign, the number of vaccines provided per day, and the fraction of vaccines supplied to each age group.

Table 4: Parameters to model the intervention to control the epidemic

Parameters	Description	Data Type
t^c_s	Timesteps when the containment measures will be applied	vector (integer)
κ_{0s}	Array of level of confinement. (Decreases population mobility. Decreases average number of contacts. Increases household isolation)	Time-varying
ϕ_s	Array of permeabilities of confined households. (Mixing among households. Decreases household isolation)	vector float (1xg)
δ_s	Array of social distancing measures. (Reduces contacts of the non-confined population)	vector float (1xg)
% of vaccines per day	% of vaccines that are supplied at each step	scalar (float)
Start vaccination	The simulation step in which vaccines are supplied	scalar (integer)
Duration	Number of simulation steps in which vaccines are supplied	scalar (integer)
ϵ^g	Fraction of vaccinations per age group	vector float (1xg)

In this scenario, we are interested in finding interventions that minimize the number of fatalities, ICUs as well as the peak in cases. We can also model the economic cost of lockdown and use this as an additional cost. In addition, for the mobility reduction intervention, we are planning to introduce penalties that account in the detrimental impact that mobility reduction can have on the economic activities. Moreover, we are also considering the use of multi-objective optimization and Pareto optimality to optimize different criteria such as the trade-off between the effectiveness of lockdowns and its detrimental impact in economic activity.

For finding effective strategies we will implement a workflow that combines optimization techniques, interactive learning and visual analytics in a similar way as it is done for the parameter calibration. Reinforcement Learning (RL) will also be considered as a solution.

3.3 Maritime Use-Case

The Maritime Use Case aims to develop solutions that enable early detection and forecasting of maritime events for safe maritime operations and efficient action planning and decision making. Based on the requirements elicitation for maritime complex critical events (D2.1), two components are developed focusing on the mitigation of vessel collision events and the avoidance of hazardous weather conditions at sea:

1. Collision Forecasting and Rerouting
2. Hazardous Weather Rerouting

The simulator of the Maritime Use Case will allow end-user operators, to simulate vessel positions and events to explore and optimize vessel navigation under the collision avoidance and hazardous weather routing focus cases, by ingesting simulated (artificial) vessel positional data and by allowing end-users to assess different alternatives through the calibration critical operational parameters. The simulated events will be presented to the end users via GUI for further evaluation and testing.

In the context of T4.2, firstly, the operational parameters for the calibration of the algorithms that were developed for both software components of the Maritime Use Case have been identified. In a second step, critical operational parameters have been selected for each focus case based on their significance in resolving the respective critical event. Based on these parameters, the specifications of the interactive learning scenarios were derived, as described in **Table 5**. To this end, we will apply the Active Learning and Optimization workflows, as they are described in the next section (Sec. 3.4).

Table 5: Interactive simulation scenario specifications for the Maritime Use Case components

Focus Case	Interactive Learning Scenario	Description
Collision Forecasting and Rerouting	Speed adjustment	Resolve the forecasted collision detection event between two vessels with a different target speed other than the current vessel speed. In this way different trajectories for the collision avoidance may be generated and different strategies can be assessed by the end user operator and consider factors that are out of scope for the collision avoidance algorithm. These factors may include operational factors that are decisive for the vessel speed, speed limitations that are not captured by the AIS and/or any other vessel sensors, space limitations, the avoidance of

		specific nearby sea areas while performing a collision avoidance manoeuvre etc.
Hazardous Weather Routing	Departure time	Evaluate different strategies based on the departure time and the changing weather conditions along the route. Under certain circumstances that include very adverse weather conditions, the route towards a destination might be blocked for several days or the alternative route might not be optimal due to the significantly increased ETA. Waiting out for weather conditions to improve is a common routing strategy of vessels to optimize maritime operations, fuel consumption and operational efficiency
Hazardous Weather Routing	Route planning with and without weather routing	Evaluate the effect on ETA when considering weather conditions in route planning. In this context, vessel crews and fleet managers will be able to quantify the effects of the hazardous weather routing component on the vessel ETA to a specific destination. This aims to improve understandability and support the respective decision makers to take informed decisions for optimal maritime operations.

Until M18, the collision forecasting and rerouting interactive speed adjustment interactive simulation functions has been implemented on the frontend. The external user is able to review collision forecasts and simulate different COLREG compliant collision avoidance paths by adjusting the vessel's target speed to a user defined value.

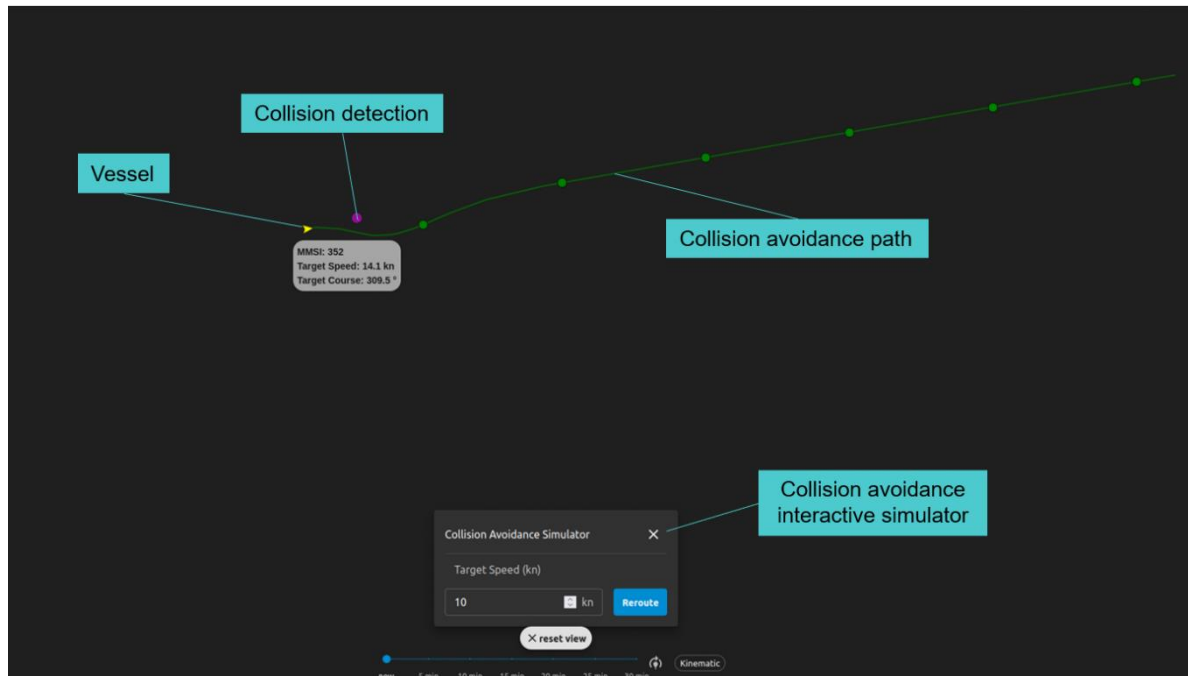


Figure 23: Vessel collision avoidance interactive simulator

3.4 Methods for the Exploration of Simulations Parameter Space

In this part we briefly present the algorithms that are going to be used for the purposes of T4.2. Apart from calibrating the simulators, recall that we also want to explore the effectiveness of interventions (or intervention policies) for example, at which crossroads to place water barriers in a city, when to apply vaccination campaigns, or how to reroute a vessel to reduce cruising costs. The values that parametrize the interventions, as well as simulator-specific configurations, form an unexplored multidimensional space of parameter values. T4.2 seeks to devise methods for effectively exploring such spaces and informing the end-user about the results in a comprehensive manner. Specifically, we will rely on (i) Active Learning [43] [44] for characterizing parameter spaces, (ii) Optimization [41] [42] for estimating the parameter values that lead to the most desired simulation outcomes, (iii) Early Time-series Classification [45] approaches for the early termination of non-relevant simulations and time-consuming simulations, and (iv) Reinforcement Learning for estimating effective policies for interventions. The corresponding methods are selected according to the requirements of each scenario.

Active Learning

To explore the parameter space, we will rely on an approach that incorporates Active Learning and has been proven effective in a similar application in the past [44]. According to this method, the parameter values are considered as vectors of random values, always within specific ranges that have been determined by domain experts. Each vector is evaluated by initiating simulations that are configured with the respective parameter values, and by assessing the simulation outcome. The assessment can either be the classification of a particular simulation outcome, or a well-defined score metric that encodes the quality of the simulation, i.e. with respect to a desired outcome, or the plausibility of the simulation. Note also that in some cases, as e.g. in the multiscale simulations use-case of CREXDATA,

the evaluation can be quite time-consuming, because of the simulator's complexity. Thus, it is not realistic to perform exhaustive search.

To that end, the method that we will adopt utilizes a Random Forest (RF) classifier to create a mapping of the parameter space based on some class labels. In the first algorithm iteration a few random input vectors are generated, and then the corresponding simulations are initiated. The data of these first simulation runs (e.g. vectors of parameter values coupled with class labels) are used to train the RF. Since the classifier is trained using only a few samples, we expect to have regions for which it is highly uncertain.

By sampling the most uncertain regions and by clustering the points included therein, we come up with only a few points as cluster centres, which are subsequently evaluated by invoking again the simulator. This process is repeated until a termination criterion is met, e.g. we reach a maximum number of iterations.

As the iterations progress, the RF fits better to the unexplored space, and thus when the algorithm terminates, we expect to have an accurate mapping of the interesting/relevant and non-interesting/non-relevant simulation instances. This trained model can then be incorporated into other workflows that are relevant to each scenario, i.e. visual analytics.

Optimization

Frequently, apart from characterizing the unexplored parameter space, it is desirable to obtain optimized parameter values, e.g. those that are very close to an outcome that the end-user seeks to obtain, or others that minimize some kind of cost. For this purpose, we can rely on Genetic Algorithms (GA) [41], Covariance Matrix Adaptation (CMA) [42], Bayesian Optimization (BO) [46], or other similar methods.

Early Time-Series Classification (ETSC)

Recall that, in some cases, the simulations might require a significant amount of time to complete. Also, in some scenarios (e.g. during optimization) it is highly probable that many non-interesting or non-relevant simulation instances will be initiated, and thus significant amounts of computational resources are spent in vain. To overcome this issue, we incorporate Early Time-Series Classification algorithms that can find the earliest time-point of a time-series at which a reliable prediction regarding its class label can be made. If a time-series is predicted to belong to any of such simulations, then the corresponding simulation can be terminated early on in time, thus freeing up resources.

In parallel, several ETSC methods have been proposed in the literature, however there is a lack of an experimental evaluation and comparison framework tailored to this domain. Also, empirical results illustrate that not every ETSC algorithm is suitable for all application domains [45]. To that end, in the context of CREXDATA we developed an open-source framework for evaluating ETSC algorithms,¹² which contains a wide spectrum of methods and appropriate datasets. We also proposed a method for ETSC that relies on state-of-the-art algorithms for full time-series classification, as this arsenal is significantly larger than the pure ETSC algorithms.

¹² <https://github.com/xarakas/ETSC>

To illustrate ETSC algorithms applicability, we performed a set of preliminary experiments, using data from the CRAWDATA's Multiscale Simulations sub-use-case. Data consists of time-series of cell category counts throughout the course of simulations with different oxygen levels, that is (i) the total number of cells, (ii) the total number of Epithelial cells, (iii) the Alive Epithelial cells, (iv) the Apoptotic Epithelial cells, (v) the Necrotic Epithelial cells, and (vi) the Infected Epithelial cells. The class labels are two, (a) the patient being in a non-critical condition, and (b) the patient being in a critical condition. Our ultimate goal is to detect critical condition simulations during their course as early as possible and terminate these instances to free up resources to give space for the exploration of non-critical condition cases.

To create the dataset, we performed a series of 1000 PhysiBoSS runs with different configuration parameters, which resulted to 356 cases of non-critical patient condition and to 644 of critical condition. Considering the categorization of the datasets in [45], this particular case is multivariate, large, and imbalanced. Taking this into account and combining also the results from our empirical evaluation of ETSC algorithms to other application domains, we recognized four of them as being the more promising with respect to ETSC-oriented evaluation measures, such as accuracy, earliness, the harmonic mean between earliness and accuracy, as well as training and testing times. Average results from a 5-fold cross validation are shown in **Figure 24**.

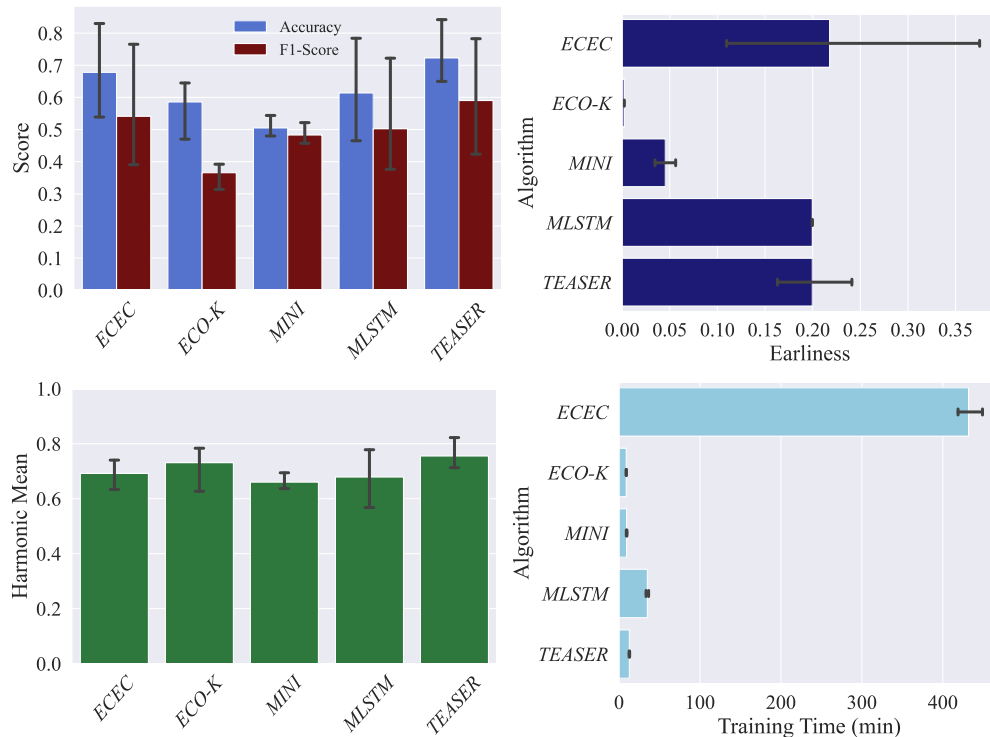


Figure 24: Preliminary results of the ETSC algorithms applied on the Multiscale Simulations CREXDATA sub-use-case

We can see that the best accuracy, F1-score, and harmonic mean are achieved by the TEASER algorithm. Although ECEC's performance is close enough for these metrics, we can observe that it requires significantly higher training times. ECO-K and MINI perform

better in terms of earliness, however their lower accuracy and F1-score values do not render them reliable for being applied in this case.

Reinforcement Learning

Additionally, we will develop a Reinforcement Learning (RL) approach for the epidemiological scenario, where an agent learns its decision-making policy by engaging with an environment itself. Through taking actions within the environment, the agent receives feedback in the form of rewards or penalties, aiding it in determining the most favourable actions across varying situations. This process allows the agent to iteratively improve its decision-making abilities. We are currently developing the different component of the RL that are described below.

Environment: The external system within which the agent acts. It responds to the actions taken by the agent (changes states) and provides feedback in the form of rewards or penalties. For us, this is the MMCA-Covid19 simulations.

State: A representation of the current situation or configuration of the environment. The agent perceives the state and selects actions based on it. Currently, it is not very clear how the States will be represented.

Action: The decision made by the agent based on the current state. Actions can lead to transitions to new states and affect the rewards received. In our case, the actions correspond to interventions being applied or retracted.

Reward: A numerical value provided e.g. by the environment to indicate the desirability of the action taken by the agent in a particular state. The goal of the agent is to maximize the cumulative reward over time. For us this could be a combination of the new cases, hospitalizations, deaths, economic impact, etc.

Interactivity capabilities for each use-case will be investigated in the following months of CREXDATA, also in synergy with “T5.2 Visual Analytics for Supporting XAI”, and “T5.3 Visual Analytics for Decision-Making Under Uncertainty”.

4 Federated Machine Learning

Distributed Deep Learning (DDL) has emerged as an alternative paradigm to the traditional centralized approach [[47], [48]] offering efficient handling of large-scale data across multiple worker-nodes, enhancing the speed of training Deep Learning (DL) models and paving the way scalable and resilient DL applications [[49] , [50], [51]].

Similarly, Federated Machine Learning (FML) builds upon the principles of DDL but takes a different approach [52]: “It enables Machine Learning on distributed data by moving the training to the data, instead of moving the data to the training”. Thus, FML enhances privacy, remains compliant with data regulations, and mitigates risks associated with centralizing data storage.

We will apply the algorithms described in this section to the Health Use Case and the Weather Emergency Use Case, as outlined in D2.1. Specifically, in the epidemiological scenario of the Health Use Case, the algorithms will be considered in the parameter calibration and optimal intervention design process. In the Weather Emergency Use Case, the algorithms will be considered in various sub-scenarios, such as guided data acquisition (EM_UC_12), object detection (EM_UC_21), and smart sensing (EM_UC_30), among others.

4.1 Related Work for Distributed & Federated Learning and Drawbacks

The traditional DDL methods typically proceed by some local training, followed by parallel sampling of local stochastic gradients at each worker. In the bulk synchronous parallel (BSP) approach [53], the gradients are averaged, and then each worker’s solution is updated using this average gradient in the Stochastic Gradient Descent (SGD) step [[54], [55]]. However, a critical challenge inherent in these traditional techniques becomes evident during each training iteration: the aggregation of gradients of DL models — often encompassing millions to many billions of parameters — across workers introduces a significant *communication bottleneck*, thereby restricting system scalability [56]. This leads to a low computation-to-communication ratio [[57], [58]], resulting in inefficient hardware utilization. Furthermore, limited bandwidth in wide-area networks significantly hinders collaborative training among remotely connected workers [59]. Addressing the communication overhead to expedite DDL algorithms has been a focal point of research for several years; speeding-up SGD is arguably the single most impactful and transformative problem in machine learning [60].

In this context, strategies that aim to alleviate the communication burden can be grouped into two main categories: (1) communication-efficient Local-SGD, where we perform several local updates on workers before aggregation (e.g., Federated Averaging – FedAvg [47]), and (2) deployment of accelerated SGD methods in DDL, since faster convergence naturally translates to fewer communication rounds (e.g., FedAdam [61] extends Adam and FedAvgM [62] extends SGD with momentum).

Problem formulation. Consider the distributed training of deep neural networks over multiple workers [[63], [64]]. In this setting, each worker has access to its own set of training data, and the collective goal is to find a common model that minimizes the overall training loss. This scenario can be effectively modelled as a distributed parallel non-convex optimization problem, formulated as follows:

$$\min F(w)_{w \in \mathbb{R}^d} \triangleq \frac{1}{K} \sum_{k=1}^K F_k(w) \quad (1)$$

Here, K is the number of workers and $F_k(w) \triangleq \mathbb{E}_{\zeta_k \sim D_k} [l_k(w; \zeta_k)]$ is the local objective function for worker k with associated data distribution D_k . The local function $l(w; \zeta_k)$ represents the loss of the data sample ζ_k given model parameter w .

Shortcomings in contemporary solutions. Communication-efficient DDL algorithms (e.g., FedAvg, FedAvgM, etc.) require predetermined and periodic round termination schedules. They do not consider whether training progresses in a desirable way. Thus, they often prescribe unnecessary synchronizations, something that makes them unsuitable for truly communication-constraint federated environments such as the ones encountered in the Weather Emergencies Use Case.

Summary. Our work addresses critical efficiency challenges in DDL, particularly in communication-constraint environments, such as the ones encountered in Federated Learning (FL) applications. We introduce and release Functional Dynamic Averaging (FDA), a novel adaptive communication-efficient strategy in distributed deep learning. Our contributions are as follows:

1. We propose FDA, which dynamically terminates training rounds by monitoring the model variance. FDA drastically reduces communication requirements, while ensuring cohesive progress towards the shared training objective.
2. We extend TensorFlow, TensorFlow Distributed and KungFu [65] to support workflows like the ones we require for FDA (federated, and without predetermined operations on participating learners – truly dynamic).
3. We evaluate and compare FDA with other DDL algorithms through an extensive suite of unique experiments with diverse datasets, models, and tasks, requiring over 200K GPU hours. This comprehensive approach significantly reduces the stochasticity typically associated with training DNNs, thereby enabling us to draw more definitive conclusions.
4. We show that FDA effectively balances the competing demands of communication and computation costs, negating the need to compromise one for the benefit of the other - unlike the conventional trade-offs encountered in the field.
5. We show that FDA effectively balances the competing demands of communication and computation costs, negating the need to compromise one for the benefit of the other - unlike the conventional trade-offs encountered in the field.
6. We demonstrate that FDA outperforms traditional and contemporary FL algorithms by orders of magnitude in communication, while maintaining equivalent model performance.
7. We show that FDA remains largely unaffected by various data heterogeneity settings, maintaining comparable performance to the IID case.

4.2 Relation to Use Cases

The Functional Dynamic Averaging (FDA) algorithm, with its communication-efficient and **model-agnostic** nature, has significant potential across all three use cases. In the maritime domain, FDA enables efficient federation between distributed data sources (e.g. networks of sensors, ships, etc.) for forecasting hazardous situations at sea, resulting in robust and timely training that aids in critical decision-making without excessive data transmission.

For weather emergency management, FDA's communication efficiency ensures that models can be updated at near real-time with new data from disparate locations, enabling authorities and first responders to proactively respond to severe weather events, thereby minimizing impact and accelerating recovery efforts.

Similarly, in health crisis management, FDA supports the continuous, online analysis of distributed epidemic data, enabling a more rapid monitoring of epidemics and the efficacy of interventions, without the administrative burden of sharing data across authorities, thus facilitating a better management of pandemic outbreaks.

Notably, FDA's compatibility with **any** scenario where **model averaging** is applicable ensures its versatility across these diverse domains (and more), promising more effective and cutting-edge federated learning solutions.

4.3 Functional Dynamic Averaging

Notation. At each time instance t , each worker k independently maintains its own set of model parameters, denoted as $w_t^{(k)} \in \mathbb{R}^d$. Let $\Delta w_t^{(k)}$ be the parameter update vector [66] computed by some stochastic optimization algorithm (e.g., SGD, ADAM) encompassing the learning rate and relevant gradients. Then, in a typical distributed learning step, each worker k applies the following update:

$$w_{t+1}^{(k)} = w_t^{(k)} + \Delta w_t^{(k)} \quad (2)$$

For all worker-local vectors, let $\bar{w}_t = \frac{1}{K} \sum_{k=1}^K w_t^{(k)}$ represent the average model at time t and \bar{w}_{t_0} the initial model shared among the workers at the start of the current communication round. Moreover, we introduce the *drift* vector $u_t^{(k)}$, that is, the change to the local model of the k -th worker at time t since the beginning of the current round at time t_0 , and \bar{u}_t as the average drift:

$$u_t^{(k)} = w_t^{(k)} - \bar{w}_{t_0}, \bar{u}_t = \frac{1}{K} \sum_{k=1}^K u_t^{(k)} = \bar{w}_t - \bar{w}_{t_0}$$

Model Variance. The *model variance* quantifies the dispersion or spread of the worker models around the average model. At time t , it is defined as:

$$\text{Var}(w_t) = \frac{1}{K} \sum_{k=1}^K \|w_t^{(k)} - \bar{w}_t\|_2^2 = \left(\frac{1}{K} \sum_{k=1}^K \|u_t^{(k)}\|_2^2 \right) - \|\bar{u}_t\|_2^2 \quad (3)$$

This measure provides insight into how closely aligned the workers' models are at any given time. High variance indicates that the models are widely spread out, essentially drifting apart, leading to a lack of cohesion in the aggregated model. Conversely, a moderate-low variance suggests that the workers' models are closely aligned, working collectively towards the

shared objective. The variance plays a critical role in our approach, as it helps to gauge the state of the training process and make informed decisions about round termination.

Round Invariant. We introduce the *Round Invariant* (RI), a condition that bounds model variance below a threshold, Θ :

$$\text{Var}(w_t) \leq \Theta \quad (4)$$

Our algorithm continuously monitors the model variance to maintain the RI. If the variance estimate exceeds the threshold Θ at any point t , the training round immediately terminates. This triggers synchronization, where all local model weights $w_t^{(k)}$ are averaged and consolidated into the new global model \bar{w}_t .

Monitoring the RI. At this point, our focus shifts to devising a method to monitor the RI, as defined in (4). To align with the general literature, we will restate the problem of monitoring the RI using the standard distributed stream monitoring formulation. We begin by defining the *local state* for each worker k , denoted by $S_k(t) \in \mathbb{R}^p$, which encapsulates information from worker k at time t ; it is updated arbitrarily. Next, we introduce the *global state*, $S(t) \in \mathbb{R}^p$, which represents the collective state of the distributed system at time t .

$$S(t) = \frac{1}{K} \sum_{k=1}^K S_k(t) \quad (5)$$

Lastly, we define a special non-linear function $H: \mathbb{R}^p \rightarrow \mathbb{R}$, with the following property:

$$H(S(t)) \leq \Theta \Rightarrow \text{Var}(w_t) \leq \Theta \quad \forall t \quad (6)$$

Conceptually, if we define $S_t(t) = \left(\|u_t^{(k)}\|_2^2, u_t^{(k)} \right) \in \mathbb{R} \times \mathbb{R}^d$ and $H(v, x) = v - \|x\|_2^2$, then it immediately follows from (3) that property (6) is satisfied. Direct monitoring of the RI is hindered by the high dimensionality of $S_k(t)$, with d potentially ranging from millions to many billions, values commonly found in DNNs. To mitigate this communication burden, we must apply dimensionality reduction to the local drifts $u_t^{(k)}$ and identify an appropriate function H , accordingly. However, this reduction in information within local states causes RI monitoring to become approximate. Upcoming sections will detail three strategies—termed "naive", "linear", and "sketch"—each offering a different balance between communication efficiency and approximation accuracy.

Algorithm. The proposed FDA algorithm is formalized in Algorithm 1. To streamline notation, t also denotes each training round, which unfolds into three main parts:

- (1) **Broadcast:** Model w_t is distributed to all workers (Line 3)
- (2) **Local Training:** In essence, as long as we can guarantee the RI, each worker continues training concurrently (Lines 4-9). Specifically, workers perform in-parallel updates to their local models $w_t^{(k)}$ using mini-batch SGD, or some other optimization algorithm, by processing their local data (Lines 5-7). After each update, workers compute and broadcast their local states $S_k(t)$ (Line 8). Then the global state $S(t)$ is calculated, and using the function H , an estimate of the model variance is derived and compared with the threshold Θ (Line 9). Depending on this comparison, the local training stage either repeats or terminates.

- (3) **Model averaging:** This stage is triggered since we can no longer guarantee the RI, i.e., $H(S(t)) > \Theta$. At this point, all local models are communicated and then averaged to form w_{t+1} , that is, the next round's global model (Lines 10-12)

Algorithm 1 Functional Dynamic Averaging - FDA

Require: $S_k(t)$: The local state $S_k(t) \in \mathbb{R} \times \mathbb{R}^p$ with $p \ll d$

Require: $H(x)$: A function s.t. $H(S(t)) \leq \Theta \Rightarrow \text{Var}(w_t) \leq \Theta$

Require: K : The number of workers indexed by k

Require: Θ : The model variance threshold

Require: b : The local mini-batch size

1. Initialize $w_1 \in \mathbb{R}^d$
 2. **for** each round $t = 1, 2, \dots$ **do**
 3. **communicate** w_t to all workers
 4. **repeat**
 5. **for** each worker $k = 1, \dots, K$ **in parallel do**
 6. $B_k \leftarrow$ (sample a batch of size b from D_k)
 7. $w_t^{(k)} \leftarrow w_t^{(k)} - \eta \nabla l_k(w_t^{(k)}; B_k)$ \triangleright In-place
 8. **communicate** $S_k(t)$
 9. **until** $H(S(t)) > \Theta$ \triangleright As per (5)
 10. **for** each worker $k = 1, \dots, K$ **in parallel do**
 11. **communicate** $w_t^{(k)}$
 12. $w_{t+1} \leftarrow \frac{1}{K} \sum_{k=1}^K w_t^{(k)}$
-

4.3.1 Linear Approximation

In the linear approach, we reduce the $u_t^{(k)}$ vector to a scalar, that is, $\langle \xi, u_t^{(k)} \rangle \in \mathbb{R}$ where $\xi \in \mathbb{R}^d$ is any unit vector. We define:

$$S_k(t) = \left(\|u_t^{(k)}\|_2^2, \langle \xi, u_t^{(k)} \rangle \right) \in \mathbb{R} \times \mathbb{R}, \|\xi\|_2 = 1$$

$$H(v, x) = v - x^2$$

Then, the condition $H(S(t)) \leq \Theta$ implies $Var(w_t) \leq \Theta$. Furthermore, a random choice of ξ is likely to perform poorly (terminate a round prematurely), as it is likely to be close to orthogonal to \bar{u}_t . A good choice would be a vector correlated to \bar{u}_t . A heuristic choice is to take \bar{u}_{t_0} , i.e., the change vector right before the current round started. All workers can estimate this without communication cost as the difference of the models at the beginning of the current and previous rounds:

$$\xi = \frac{w_{t_0} - w_{t-1}}{\|w_{t_0} - w_{t-1}\|_2}$$

4.3.2 Sketch Approximation

An optimal estimator for $\|\bar{u}_t\|_2^2$ can be obtained through the utilization of AMS sketches, as detailed in [67]. An AMS sketch of a vector $v \in \mathbb{R}^d$ is an $l \times m$ real matrix Ξ :

$$sk(v) = \Xi = [\xi_1 \xi_2 \dots \xi_l]^T \in \mathbb{R}^{l \times m}, l \cdot m \ll d$$

The $sk(\cdot)$ operator is linear and can be computed in $O(l \cdot d)$ steps. Let $l = O\left(\log \frac{1}{\delta}\right)$ and $m = O\left(\frac{1}{\epsilon^2}\right)$. The function

$$M_2(\Xi) = \text{median}\{\|\xi_i\|_2^2, i = 1, \dots, l\}$$

Is an excellent estimator of $\|v\|_2^2$: with probability at least $(1 - \delta)$, $M_2(sk(v))$ is within ϵ -relative error of $\|v\|_2^2$. Consequently, we define:

$$S_k(t) = \left(\|u_t^{(k)}\|_2^2, sk(u_t^{(k)}) \right) \in \mathbb{R} \times \mathbb{R}^{l \times m}$$

$$H(v, \Xi) = v - \frac{1}{1 + \epsilon} M_2(\Xi)$$

Then, the condition $H(S(t)) \leq \Theta$ implies $Var(w_t) \leq \Theta$ with probability at least $(1 - \delta)$.

4.4 Extending TensorFlow and KungFu to support FDA workflows

4.4.1 TensorFlow Distributed

TensorFlow provides an API to facilitate training distribution across multiple GPUs, machines, or TPUs, accommodating various use cases. **MirroredStrategy** supports synchronous distributed training on multiple GPUs within a single machine. It creates one replica of the model per GPU, with each variable mirrored across all replicas i.e., every replica keeps its own local copy of every variable.

MultiWorkerMirroredStrategy extends the capabilities of **MirroredStrategy** to support synchronous distributed training across multiple workers, each potentially equipped with multiple GPUs. Like **MirroredStrategy**, it creates copies of all model variables on each device across all workers, ensuring synchronization across all replicas.

Typically, MultiWorkerMirroredStrategy executes a single training step on a batch of data for one replica and aggregates gradients across all replicas before applying them. With our necessary extensions to MultiWorkerMirroredStrategy, we apply gradients locally without the default aggregation and later average the model weights, instead, when the RI can no longer be guaranteed. Our implementation in TensorFlow supports both systems with the Slurm workload manager and custom clusters (specified through IP and Port information).

4.4.2 KungFu

KungFu [65] is a distributed machine learning library for TensorFlow. Its focus is to provide fast inter-GPU communication making use of the Nvidia Collective Communications Library (NCCL). NCCL is following on the Message Passing Interface (MPI) and achieves peer-to-peer communication by designing a bidirectional ring topology based on the actual topology of the network's GPUs. KungFu also offers custom tree topologies for the network of clients. This custom topology setting proved useful in simulating challenging network conditions prominent in Federated Learning scenarios.

We extend KungFu to be an easily deployable implementation of Functional Dynamic Averaging (FDA) and an alternative to Distributed TensorFlow. The standard periodic synchronization scheme was transformed to a dynamic one.

The experiments were also executed on a High-Performance Computing (HPC) setting using the Slurm Workload Manager. For a detailed analysis of the work done on KungFu please refer to the corresponding integrated master thesis [68].

4.4.3 Simulations with TensorFlow

Utilizing TensorFlow Distributed or KungFu entails incurring communications costs between workers, thereby slowing down our extensive suite of experiments by a lot. To mitigate this, we developed a package to simulate the DDL, via FDA, without actual inter-worker communication. A detailed exposition of this package can be found in [69].

4.5 Experiments & Comparison to Prior Work

Table 6: Setup

Neural Network	d	Dataset	Θ	b	K	Optimizer	Algorithms
LeNet-5	62K	MNIST	{0.5, 1, 1.5, 2, 3, 5, 7}	32	{5, 10 ... , 60}	Adam	FDA, Synchronous, FedAdam
VGG16*	2.6M	MNIST	{20, 25, 30, 50, 75, 90, 100}	32	{5, 10 ... , 60}	Adam	FDA, Synchronous, FedAdam
DenseNet121	6.9M	CIFAR-10	{200, 250, 275, 300, 325, 350, 400}	32	{5, 10 ... , 60}	SGD-NM	FDA, Synchronous, FedAvgM
DenseNet201	18M	CIFAR-10	{350, 500, 600, 700, 800, 850, 900}	32	{5, 10 ... , 60}	SGD-NM	FDA, Synchronous, FedAvgM
ConvNeXtLarge (fine-tuning)	197M	CIFAR-100	{25, 50, 100, 150}	32	{3, 5}	AdamW	FDA, Synchronous

--	--	--	--	--	--	--	--

Datasets & Models. The core experiments involve training Convolutional Neural Networks (CNNs) of varying sizes and complexities from scratch on two datasets: MNIST [70] and CIFAR-10 [71]. For the MNIST dataset, we employ LeNet-5 [72], composed of approximately 62 thousand parameters, and a modified version of VGG16 [73], denoted as VGG16*, consisting of 2.6 million parameters. VGG16* was specifically adapted for the MNIST dataset, a less demanding learning problem compared to ImageNet [74], for which VGG16 was designed. In VGG16*, we omitted the 512-channel convolutional blocks and downsampled the final two fully connected (FC) layers from 4096 to 512 units each. Both models use Glorot uniform initialization [75]. For CIFAR-10, we utilize DenseNet121 and DenseNet201 [76], as implemented in Keras [77], with the addition of dropout regularization layers at rate 0.2 and weight decay of 10^{-4} , as prescribed in [76]. The DenseNet121 and DenseNet201 models have 6.9 million and 18 million parameters, respectively, and are both initialized with He normal [78]. Lastly, we explore a transfer learning scenario on the CIFAR-100 dataset [71], a choice reflecting the DL community's growing preference of using pre-trained models in such downstream tasks [[79], [80]]. For example, a pre-trained visual transformer (ViT) on ImageNet, transferred to classify CIFAR-100, is currently on par with the state-of-the-art results for this task [81]. We adopt this exact transfer learning scenario, leveraging the more powerful ConvNeXtLarge model, pre-trained on ImageNet, with 198 million parameters [77]. Following the feature extraction step [66], the testing accuracy on CIFAR-100 stands at 60%. Subsequently, we employ and evaluate our FDA algorithms in the arduous fine-tuning stage, where the entirety of the model is trained [82].

Algorithms. We consider five distributed deep learning algorithms: LinearFDA, SketchFDA, Synchronous, FedAdam [61], and FedAvgM [62]; the first three are standard in all experiments. Depending on the local optimizer, Adam [83] or SGD with Nesterov momentum (SGD-NM) [84], we also include their federated counterparts FedAdam or FedAvgM, respectively. Notably, Synchronous was derived from the Bulk Synchronous Parallel approach; can be understood as a special case of the FDA Algorithm 1 where Θ is set to zero.

Evaluation Methodology. Comparing DDL algorithms is not straightforward. For example, comparing DDL algorithms based on the average cost of a training epoch can be misleading, as it does not consider the effects on the trained model's quality. To achieve a comprehensive performance assessment of FDA, we define a *training run* as the process of executing the DDL algorithm under evaluation, on (a) a specific DL model and training dataset, and (b) until a final epoch in which the trained model achieves a specific *testing accuracy* (termed as *Accuracy Target* in the Figures). Based on this definition, we focus on two performance metrics:

1. *Communication cost*, which is the total data (in bytes) transmitted by all workers. Translating this cost to *wall-clock time* (i.e., the total time required for the computation and communication of the DDL) depends on the network infrastructure connecting the workers. Its impact is larger in FL scenarios, where workers often use slower Wi-Fi connections.
2. *Computation cost*, which is the number of mini-batch steps (termed as *In-Parallel Learning Steps* in the Figures) performed by each worker. Translating this cost to

wall-clock time is determined by the mini-batch size and the computational resources of the nodes. Its impact is larger for nodes with lower computational resources.

Hyper-Parameters & Optimizers. Hyper-parameters unique to each training dataset and model are detailed in **Table 6**; Θ is pertinent to FDA algorithms and not applicable to others. Notably, a guideline for setting the parameter Θ is provided in Section 3.4.2. For experiments involving FedAvgM and FedAdam, we use $E = 1$ local epochs, following [61]. For experiments with LeNet-5 and VGG16*, local optimization employs Adam, using the default settings as per [35]. In these cases, FedAdam also adheres to the default settings for both local and server optimization [[77], [61]]. For DenseNet121 and DenseNet201, local optimization is performed using SGD with Nesterov momentum (SGD-NM), setting the momentum parameter at 0.9 and learning rate at 0.1 [76]. For FedAvgM, local optimization is conducted with default settings [[77], [62]], while server optimization employs SGD with momentum, setting the momentum parameter and learning rate to 0.9 and 0.316, respectively [61]. Lastly, for the transfer learning experiments, local optimization leverages AdamW [85], with the hyper-parameters used for fine-tuning ConvNeXtLarge in the original study [86].

Data Distribution. In all experiments, the training dataset is divided into approximately equal parts among the workers. To assess the impact of data heterogeneity, we explore three scenarios: (1) independent and identically distributed (IID) setup, (2) percentage-wise non-IID setup: a portion of the dataset is sorted and sequentially allocated to workers with the remainder distributed in an IID fashion, and (3) label-specific non-IID setup: assignment of all samples from a specific label to a few workers while distributing the rest in an IID fashion, introducing label-centric data concentration. All three scenarios of data heterogeneity are evaluated using the same hyper-parameters. A constant aspect across our various experiments, for a specific data heterogeneity setting, is the data partitioning, solely based on the number of workers involved. For example, in the IID setting, for all tests involving $K = 25$ workers, data partitioning remains consistent.

4.5.1 Results

Due to the extensive set of unique experiments (over 1000), as detailed in **Table 6**, we leverage Kernel Density Estimation (KDE) plots [87] to visualize the bivariate distribution of computation and communication costs incurred by each strategy for attaining the *Accuracy Target*. These KDE plots provide a high-level overview of the cost trade-off for training accurate models. As an illustrative example, Figure 25 depicts the strategies' bivariate distribution for the LeNet-5 model trained on MNIST with different data heterogeneity setups. In these plots, the SketchFDA distribution is generated from experiments across all hyper-parameter combinations (Θ and K in Table 1) that attained the *Accuracy Target* of 0.985.

FDA balances Communication vs. Computation. DDL algorithms face a fundamental challenge: balancing the competing demands of computation and communication. Frequent communication accelerates convergence and potentially improves model performance, but incurs higher network overhead, an overhead that may be prohibitive when workers communicate through lower speed connections. Conversely, reducing communication saves bandwidth but risks hindering or even stalling convergence. Traditional DDL approaches, like Synchronous, require synchronizing model parameters after every learning step, leading to significant communication overhead but facilitating faster convergence (lower computation cost). This is evident in Figure 25, Figure 26, and Figure 27 (where Synchronous appears in the bottom right --- low computation, very high communication). Conversely, Federated

Optimization (FedOpt) methods [61] are designed to be very communication-efficient, drastically reducing communication between devices (workers) at the expense of increased local computation. Indeed, as shown in the aforementioned figures, FedAvgM and FedAdam reduce communication by orders of magnitude but at the price of a corresponding increase in computation. Our two proposed FDA strategies achieve the best of both worlds: the low computation cost of traditional methods and the communication efficiency of FedOpt approaches, as in Figure 25, Figure 26, and Figure 27. In fact, they significantly outperform FedAvgM and FedAdam in their element, that is, communication-efficiency. Across all experiments, the FDA methods' distributions lie in the desired bottom left quadrant --- low computation, very low communication.

FDA counters diminishing returns. The phenomenon of *diminishing returns* states that as a DL model nears its learning limits for a given dataset and architecture, each additional increment in accuracy may necessitate a disproportionate increase in training time, tuning, and resources [[66], [88]]. We first clearly notice this with VGG16* on MNIST in Figure 26 for all three data heterogeneity settings. For a 0.001 increase in accuracy (effectively 10 misclassified testing images) FedAdam needs an order of magnitude more computation and communication. Similarly, Synchronous requires comparable increases in computation and approximately half an order of magnitude more in communication. On the other hand, the FDA methods suffer a slight (if any) increase in computation and communication for this accuracy enhancement. For DenseNet121 and DenseNet201 on CIFAR-10 (Figures omitted due to space constraints), FedAvgM and Synchronous require half an order of magnitude more computation and communication to achieve the final marginal accuracy gains (0.78 to 0.81 for DenseNet121, and 0.78 to 0.8 for DenseNet201). In contrast, the FDA methods have almost no increase in communication and comparable increase in computation.

FDA is resilient in data heterogeneity. In DDL, data heterogeneity is a prevalent challenge, reflecting the complexity of real-world applications where the IID assumption often does not hold. The ability of DDL algorithms to maintain consistent performance in the face of non-IID data is a critical metric for their effectiveness and adaptability. Our empirical investigation reveals the FDA methods' noteworthy resilience in such heterogeneous environments. For LeNet-5 on MNIST, as illustrated in Figure 25, the computation and communication costs required to attain a test accuracy of 0.985 show negligible differences across the IID and the two Non-IID settings (Label "0", 60%).

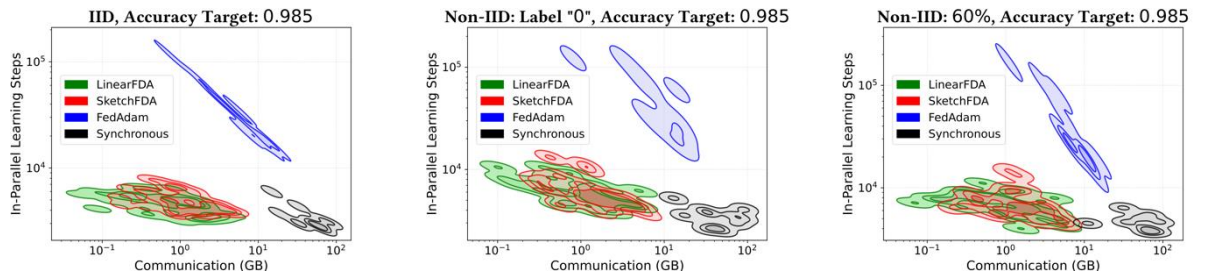


Figure 25: LeNet-5 on MNIST

Similarly, for VGG16* on MNIST, Figure Figure 26 demonstrates that achieving a test accuracy of 0.995 incurs comparable computation and communication costs across the IID and the two Non-IID settings (Label "0", Label "8"); while overall costs are aligned, the

distributions of the computation costs exhibit greater variability, yet remain closely consistent with the IID scenario.

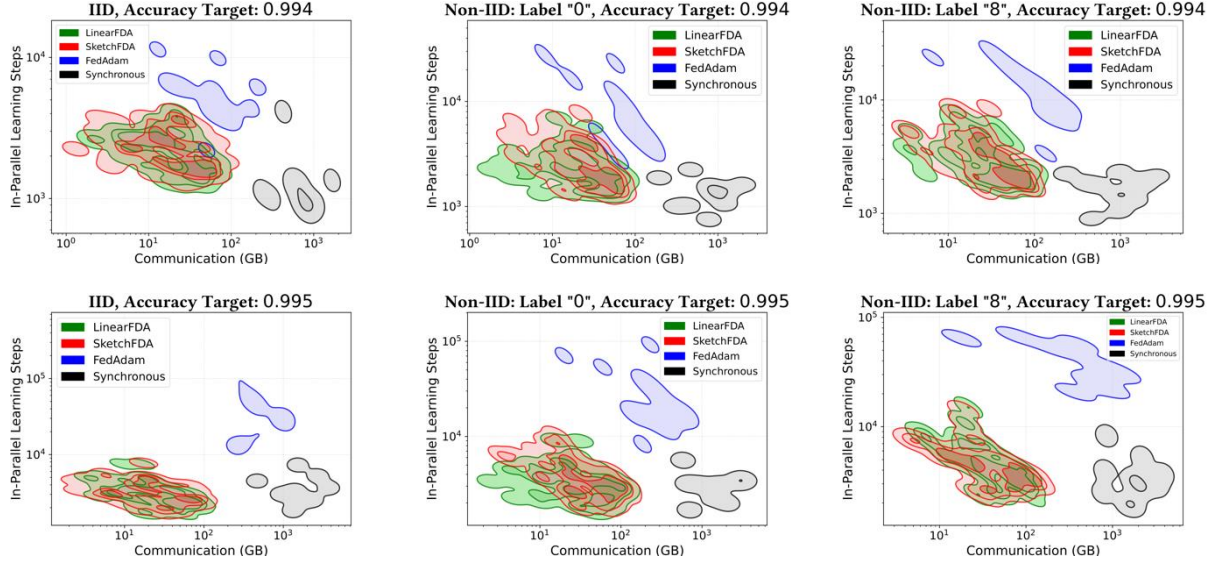


Figure 26: VGG16* on MNIST

FDA generalizes better. The factors determining how well a DL algorithm performs are its ability to: (1) make the training accuracy high, and (2) make the gap between training and test accuracy small. These two factors correspond to the two central challenges in DL: underfitting and overfitting [66]. For DenseNet121 on CIFAR-10, with a test accuracy target of 0.8, as illustrated in Figure 28, Synchronous and FedAvgM exhibit overfitting, with a noticeable discrepancy between training and test accuracy. In stark contrast, the FDA methods have an almost zero accuracy gap. Turning our focus to DenseNet201 on CIFAR-10, with a test accuracy target of 0.78, Synchronous again tends towards overfitting, while FedAvgM shows a slight improvement but still does not match the FDA methods, which continue to exhibit exceptional generalization capabilities, evidenced by a minimal training-test accuracy gap (Figure 26). Notably, given the necessity to fix hyper-parameters Θ and K for the training accuracy plots, we selected two representative examples. The patterns of performance we highlighted are consistent across most of the tests conducted.

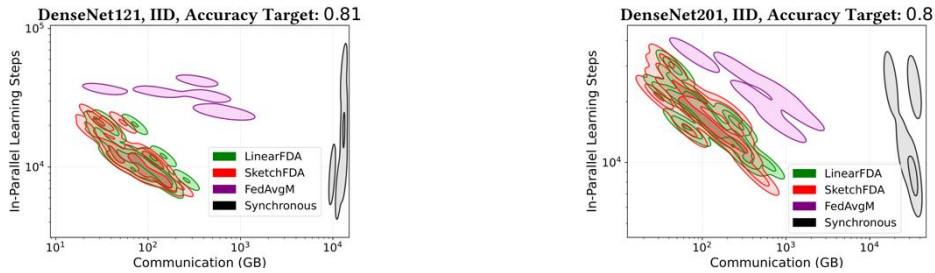


Figure 27: DenseNet121 and DenseNet201 on CIFAR-10

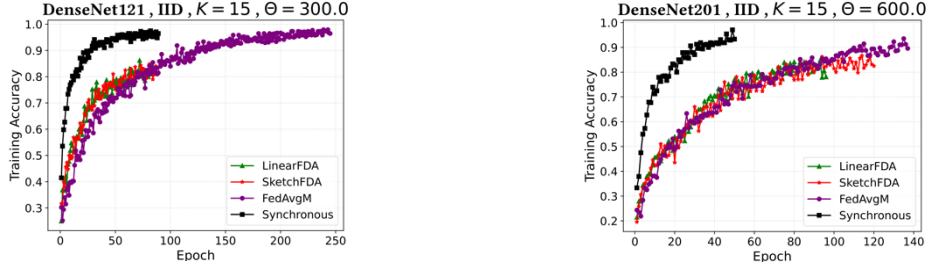


Figure 28: Training accuracy progression with a (test) accuracy target of 0.8 (left), and 0.78 (right). A smaller final gap between training and target accuracy indicates less overfitting, i.e., better generalization capabilities of the trained model

Dependence on K. In distributed computing, scaling up typically results in proportional speed improvements. In DDL, however, scalability is less predictable due to the nuanced interplay of computation and communication costs with convergence, complicating the expected linear speedup. This unpredictability is starkly illustrated with LeNet-5 and VGG16* on the MNIST dataset across all data heterogeneity settings and all strategies. Specifically, Figure 29 demonstrates (at its left part) that increasing the number of workers does not decrease computation -- except for FedAdam which begins with significantly high computation -- but rather exacerbates communication (shown at the left part of the figure). These findings are troubling, as they reveal scaling up only hampers training speed and wastes resources. However, for more complex learning tasks like training DenseNet-121 and DenseNet-201 on CIFAR-10 (Figure 30 and Figure 31), the expected behaviour emerges. Scaling up (K increase) leads to a decrease in computation cost for all strategies. Communication cost, however, increases with K for all methods except Synchronous, which maintains constant communication irrespective of worker count, but at the expense of orders of magnitude higher communication overhead. Notably, while our findings might, in some cases, suggest potential speed benefits of not scaling up (smaller K), DDL is increasingly conducted within federated settings, where there is no other choice but to utilize the high number of workers.

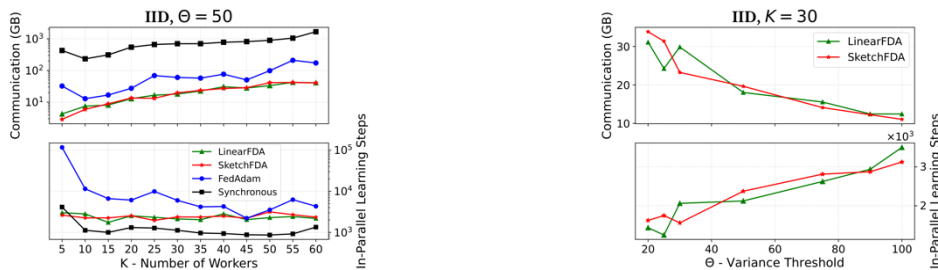


Figure 29: VGG16* on MNIST – Accuracy Target: 0.994

FDA: Dependence on Θ . The variance threshold Θ can be seen as a lever in balancing communication and computation; essentially, it calibrates the trade-off between these two costs. A higher Θ allows for greater model divergence before synchronization, reducing communication at the cost of potentially increased computation to achieve convergence.

This impact of Θ is consistently observed across all two FDA strategies, learning tasks, and data heterogeneity settings (Figure 29, Figure 30 and Figure 31). Interestingly, for more complex models like DenseNet121 and DenseNet201 on CIFAR-10, increasing the variance threshold (Θ) does not lead to a significant rise in computation cost, as illustrated in Figure 30 and Figure 31. It suggests that the FDA methods, by strategically timing synchronizations (monitoring the variance), substantially reduce the number of necessary synchronizations without a proportional increase in computation for the same model performance; this is particularly promising for complex DDL tasks.

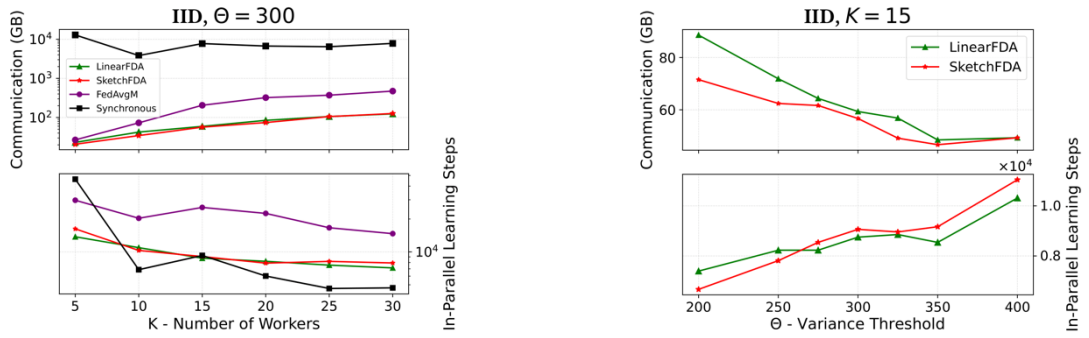


Figure 30: DenseNet121 on CIFAR-10 – Accuracy Target: 0.8

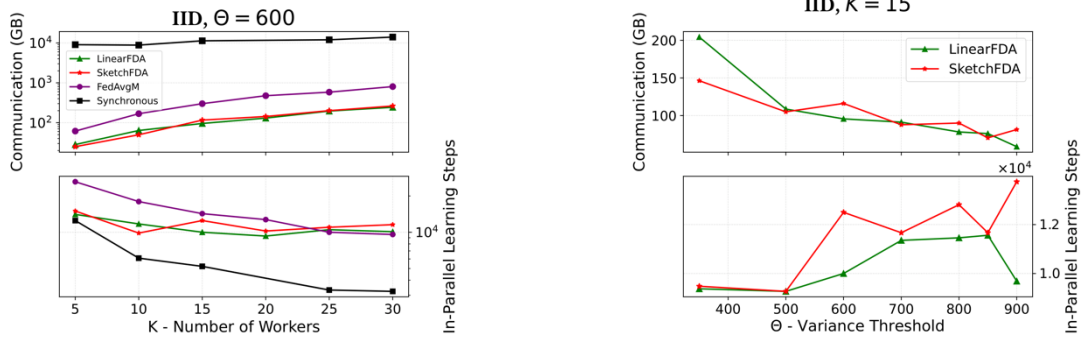


Figure 31: DenseNet201 on CIFAR-10 – Accuracy Target: 0.78

FDA: Choice of Θ . The experimental results suggest that selecting any Θ within a specific order of magnitude (e.g., between 10^2 and 10^3 for DenseNet201) ensures convergence, as demonstrated in (Figure 29, Figure 30 and Figure 31). Therefore, identifying this range becomes crucial. To this end, we conducted extensive exploratory testing to estimate the Θ ranges for each learning task which are predominantly influenced by the number of parameters d of the DNN. Within this context, Θ values outside the desirable range exhibit notable effects: below this range, the training process mimics Synchronous or Local-SGD approaches with small τ , while exceeding it leads to non-convergence.

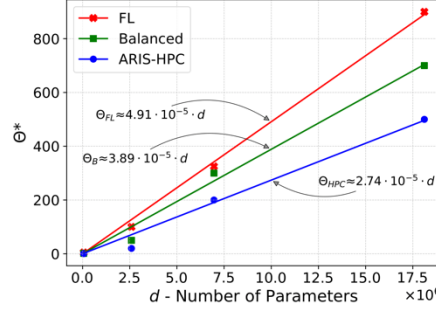


Figure 32: Empirical Estimation of the Variance Threshold

Subsequently, having identified the optimal ranges for Θ , we selected diverse values within them for our experimental evaluation (**Table 6**), thereby investigating different computation and communication trade-offs. For instance, in the ARIS-HPC environment with an InfiniBand connection (up to 56Gb/s), experiments show that training wall-time (the total time required for the computation and the communication of the DDL) is predominantly influenced by computation cost, rendering communication concerns negligible. In such contexts, lower Θ values are favoured due to their computational efficiency. On the contrary, in FL settings, where communication typically poses the greater challenge, opting for higher Θ values proves advantageous; reduction in communication achieved with higher Θ values will translate in a large reduction in total wall-time. To assist researchers in selecting an optimal variance threshold, Figure 32 presents empirical estimations for Θ across three distinct learning settings: FL (assuming a common channel of 0.5 Gbps), Balanced (communication-computation equilibrium), and HPC.

FDA: Linear vs. Sketch. In our main body of experiments, across most learning tasks and data heterogeneity settings, the two proposed FDA methods exhibit comparable performance, as illustrated in Figure 25, Figure 26, and Figure 27. This suggests that the precision of the variance approximation is not critical; occasional "unnecessary" synchronizations do not significantly impact overall performance. However, in all experiments within the more intricate transfer learning scenario, LinearFDA requires approximately 1.5 times more communication than SketchFDA to fine-tune the deep ConvNeXtLarge model to equivalent performance levels (Figure 33). In light of these findings, we conclude the following: for straightforward and less demanding tasks, LinearFDA is the recommended option due to its simplicity and lower complexity per local state computation. On the other hand, for intricate learning tasks and deeper models, SketchFDA becomes the preferred choice, particularly if communication-efficiency is paramount.

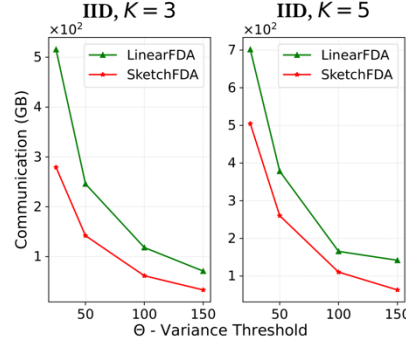


Figure 33: ConvNeXtLarge on CIFAR-100 (transfer learning from ImageNet) — Deployment of FDA during the fine-tuning stage with Accuracy Target of 0.76

FDA on various topologies. We used KungFu [65] to run similar experiments to the other two approaches (TensorFlow Simulations and Distributed TensorFlow), with similar results that showcase the effectiveness of the FDA methods in communication reduction. In addition to the other methods, using KungFu we had the opportunity to perform our experiments on different network topologies (Figure 34). Furthermore, three different topologies were used for these experiments, one being the default ring topology of KungFu and NCCL, a binary tree and a star topology. We chose to test these topologies for the maximum number of clients available as communication cost would be more prominent. Bearing in mind that communication is more expensive for the new network topologies, we assumed that the training performance of the FDA methods would be even more beneficial. This is clear in the figures above, as the FDA methods perform much better in real training time for 50 epochs. This means that they would be much needed for scenarios where the communication cost is high. For the experimental hyper-parameters please refer to the diploma thesis [68].

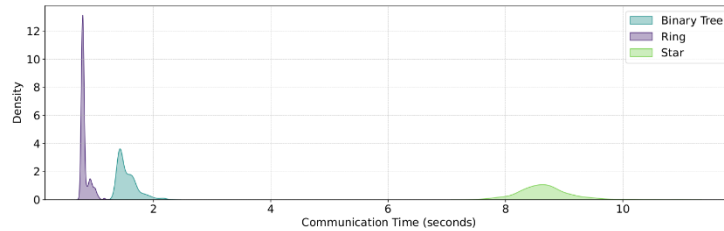


Figure 34: Communication time distribution with variable topology

4.6 FDA in NeRF computation for the Weather Emergencies Use Case

4.6.1 NeRF and distributed pipeline planning

Neural Radiance Field (NeRF) is a method that uses deep learning and can create photorealistic 3D representations of scenes. A basic NeRF model requires the input of 2D images of the scene, along with proper metadata describing the camera positioning. Our work on FDA and distributed deep learning could be used in the future to distribute the training process of NeRFs to multiple computing nodes, while also avoiding the transfer of images to a central location. With this in mind, we delved into the largely open problem of distributed NeRF training.

The metadata of the images are also called poses and are usually a 6 degrees of freedom variable depicting where the camera was located during the image capture (translation) and its viewing direction (rotation). The poses of the camera are not known a-priori, and they can be calculated using a Structure from Motion (SfM) algorithm like COLMAP [89]. These algorithms can extract these images metadata by finding and linking key-points that can be found in different images of the dataset. This SfM phase, prior to NeRF training, should also be distributed among the nodes of the network.

Now we can describe an ideal pipeline so that we can distribute the NeRF training process during the use case of a weather emergency. Let's say a weather emergency happens in a large scene and we want to deploy a group of drones to scan the area capturing 2D images. These images should ideally be transmitted to multiple GPU-equipped mobile computing centres located as close to the drones as possible. Each such server would end up having a subset of the total of images and running SfM experiments using them. The SfM camera poses located on different servers get aligned using their camera's GPS sensors data. Then each server trains a local NeRF model based on their own data. After some time, or when the FDA method indicates, all local NeRF models are transmitted in a central location where they get aggregated accordingly to a global NeRF. This global NeRF model can then be used to render novel views of the area in 3D by civil protection personnel.

4.6.2 Experimentation and limitation

To organize and perform the experiments, we have collaborated with the Deutsches Rettungsrobotik-Zentrum (DRZ) through extensive meetings. DRZ provided a dataset of 249 images taken by a DJI M30T drone, depicting their premises in Dortmund from various angles. The scene includes cars and a rubble area and resembles actual scenarios of the weather emergency use case. We aim for the resulting 3D representation to include these details in high resolution Figure 35.

We have experimented with different kinds of Structure from Motion (SfM) algorithms such as WebODM, COLMAP [89], Hierarchical Localization [90] and Pixel Perfect SfM [91]. We have concluded that COLMAP is more than enough to estimate camera poses adequately. As for NeRF models, we have mostly focused on the Nerfacto model, a model created for the NeRF framework Nerfstudio [92]. We have extended Nerfstudio to be able to conduct distributed learning simulations, using groups of images that are distributed either uniformly or in geographical blocks. The results of distributed experiments are promising, especially for multiple clients, in terms of reducing the training time required.

The main obstacle, that is currently work in progress, is improving the resulting global model. The aggregated model rendered images exhibit some "waving" effects that reduce the quality and sharpness of them. This is most likely caused by noise between the pose estimation of different SfM models.

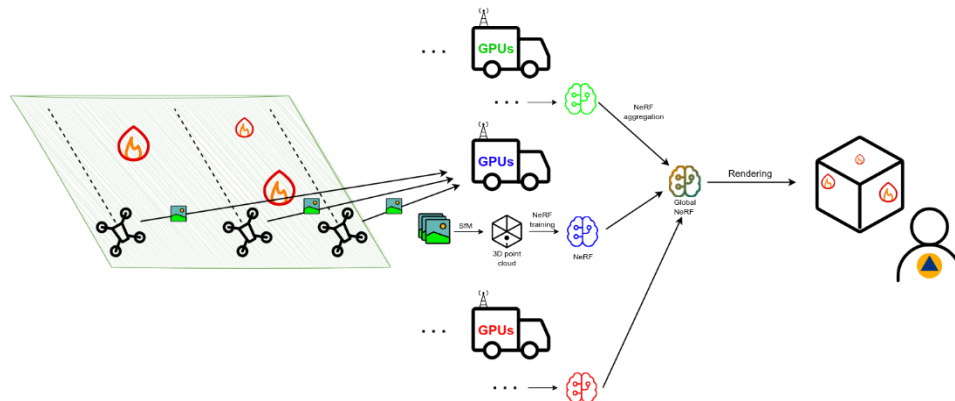


Figure 35: Experimental setup

5 Optimized Distributed Analytics as a Service

5.1 Motivation and Optimization Aspects in CREXDATA

From its very foundation, the architectural framework of CREXDATA, as described in Deliverable D3.1, operates across the cloud to edge continuum including the lot of available devices. From resource constrained sensors, of medium processing and memory capacity, to fog devices and powerful servers at the cloud side. The naïve approach of executing the prescribed extreme scale analytics only at the cloud side is not scalable as it entails a number of important disadvantages. First, naively relaying the continuously produced streaming data at the cloud side incurs extreme network load depleting the available bandwidth. This diminishes the potential for scalability in the federated setting [93] and, in turn, causes network latencies. Such latencies hinder the execution of analytics pipelines in a real-time, online fashion. Second, at the edge side of the network, data transmission is by far the main culprit in energy drain for battery powered edge devices [94]. Therefore, transmitting the raw data reduces the network lifetime and increases administrative and equipment costs. Third, all the devices across the cloud to edge continuum have some, variable and potentially limited; though existing, processing power. By naively using every node in the network only as a data producer or relay node, one does not exploit the full, aggregative processing power of the network.

Hence, the preferable way to go is to distribute the processing load of an extreme scale analytics workflow across the devices of the network based on the computational, memory demands of each operator (relational, Machine Learning, Neural Learning or User Defined Function operator) and the computational capacity of each device. However, this is easier said than done because the involved task is a combinatorial optimization resource allocation problem of interconnected tasks (operators), each of which can be executed in multiple devices; and based on the devices it chooses for itself, it affects the performance of all upstream (to which it provides input) operators of the workflow.

On the other hand, it would be error prone to allow the user or the application to manually decide on which device each operator will be executed since, in extreme scale scenarios, the potential performance of a number of concurrently executed workflows and/or the number of the devices of the network is not tractable by a human operator. Therefore, what we need for optimized streaming analytics as a service in the scope of CREXDATA is an algorithmic suite which automates the optimization process under any workflow(s) and network set up. Each algorithm of this suite receives as input a logical workflow. A **logical workflow** incorporates the application logic, but is deprived from the actual, physical execution details. The output of each algorithm is a **physical execution plan** or **physical workflow** where, for each of the operators of the logical workflow(s), it automatically prescribes the device(s) it will be executed on, along with possible execution options (discussed shortly).

Consider, for instance, **Figure 36**. On the left-hand side there is a logical workflow, while on the right-hand side, an example of physical, small-scale network is depicted. The optimization suite of CREXDATA should decide on the mapping of operators to the physical devices for execution. However, it can easily be observed that there is a large number of possible mappings per operator as well as combinations of operators.

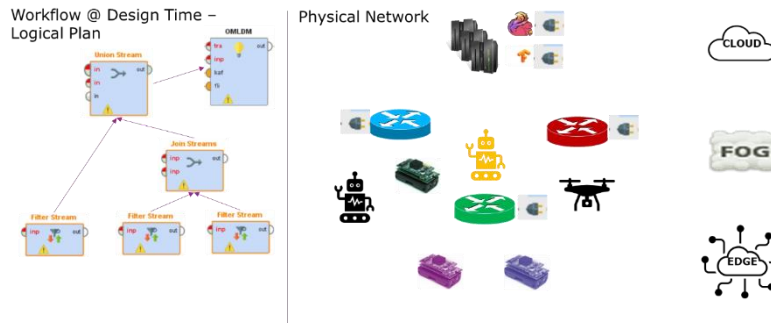


Figure 36: Logical Plan and Physical Network. The CREXDATA Optimized Analytics as a Service should device the placement of Logical Plan Operators to Physical Devices

More precisely, in the general case, a wide variety of options regarding the physical execution of an operator exist:





- i. An operator may be assigned for execution one out of the entire set or to one out of a subset of the available devices. Depending on the capacity of the device and the computational, memory demands of the operator.
- ii. An operator may be replicated to multiple devices, or be assigned to only one device. It may also get executed on a device under a certain degree of parallelism.
- iii. An operator (e.g., Neural Learning operator) may be executed in a distributed fashion at the cloud side or across the network in a federated way.
- iv. An operator may be assigned to a device choosing to prefetch or cache its data and transmit them only upon demand.
- v. An operator may send its raw data or a subset (e.g., sample) of them.

To better understand the complexity of the problem at hand, one should consider that if:

- O operators in the CREXDATA workflow(s)
- P possible options in categories (ii)-(v) per operator
- S network devices/sites (category (i) above)

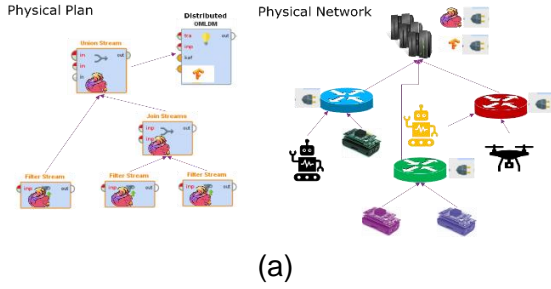
exist, an exhaustive search algorithm has a search space of $(P \cdot S)^O$ possible plans.

What is more, is that the above are not one-shot decisions, but may need to get revised as time passes. This is due to the fact that new sites may enter or depart from the network or because the statistical properties of the ingested streams and, thus, of the workload may change. This says that in a highly volatile, extreme scale, streaming setup it may not be worthwhile to invest a large amount of time in prescribing a physical workflow by the time a logical workflow is submitted, but instead, start with a good enough physical plan, monitor its performance and adjust the execution as time passes.

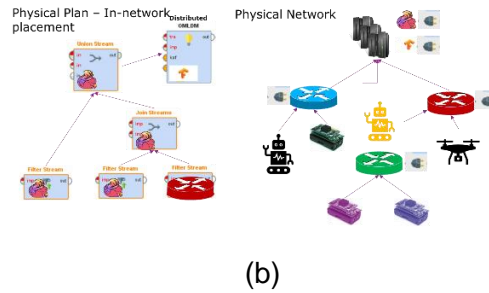
For instance, **Figure 37a** shows the evolution of the execution of a logical workflow initially deployed entirely at the cloud side (denoted by the Apache Flink icon  on each operator) at time t_0 . Then, at time t_1 , the optimizer discovers a new physical plan, stops the execution of the previous one and chooses to change the rightmost, bottom operator from  to the  site (**Figure 37b**). This operator, as is the case with all operators at the bottom of the logical workflow (Figure 36), is a filter operator. Therefore, after an instant placement at time t_0 , the optimizer sees the opportunity to distribute the load to one more site () and also

reduce the network load by filtering (choosing a subset of the produced streams) at the fog side rather than currying the raw data to the cloud.

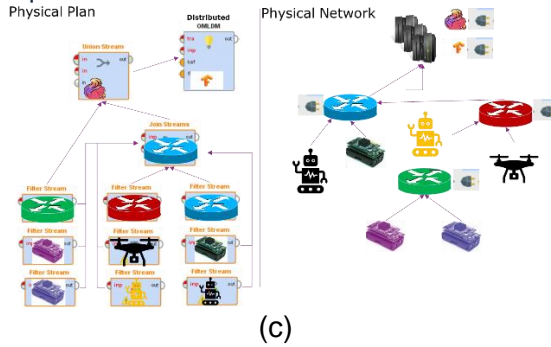
**Optimization Process Example –
@ t_0 : Initial Plan All@Cloud**



**Optimization Process –
@ t_1 : In-network placement
of 1 operator**



**Optimization Process –
@ t_{k+1} : In network placement of more
operators**



**Optimization Process –
@ t_{k+2} : Turn Distributed to Federated**

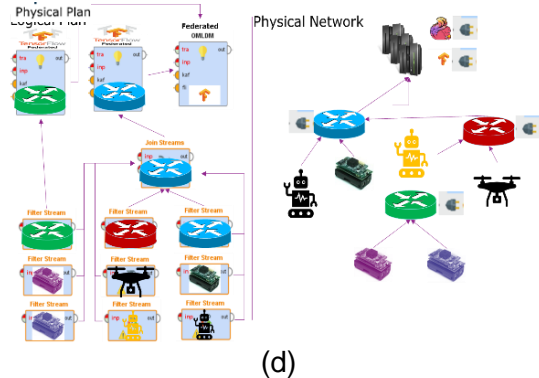





Figure 37: Different physical plans prescribed as time passes from t_0 to t_{k+2}

Continuing our running example at time t_{k+1} , **Figure 37c** shows that the optimizer has chosen to act similarly for all filter operators at the bottom of the workflow, assigning their execution to more sites at both the fog and the edge side of the network. Furthermore, each of these operators has been replicated to multiple sites in order to further reduce the processing load and restrict bandwidth consumption. For instance, the rightmost filter operator at the bottom of the workflow is now executed at a fog device , an edge – sensor device  and another, mobile edge site . Finally, **Figure 37d** shows the final execution plan devised by the optimizer where also the topmost Machine Learning operators of the workflow are chosen to get executed in a federated way.

Given the above, the optimization approach in CREXDATA accounts for arbitrarily large-scale settings including arbitrarily **many workflows**, arbitrarily **many sites**, arbitrarily **many physical execution options**, the volatility of the network setup and the data stream characteristics. The approach CREXDATA chooses to achieve these goals can be summarized in the following key optimization concepts:

- It starts with an instantly devised, empirically good enough plan,

- as the initial plan is executed, the chosen algorithm from CREXDATA algorithmic suite continues executing at the background to improve the running physical plan,
- all algorithms are inherently parallel in nature during the exploration of all or a vast amount of possible physical plans from the arbitrarily large sized search space, to achieve exploring as many candidate physical plans as possible in an sorter period of time,
- the suite includes **exhaustive** search, **sampling-based** and **greedy** algorithms trading the algorithm execution time for the goodness of the devised physical plan.

The parallel nature of CREXDATA algorithms themselves (not of the parallel execution of the workflows) is a unique feature in the relevant literature [95] [96] [97] [98] [99].

5.2 Optimization Setup

CREXDATA optimizes logical workflows devising physical plans based on the following performance criteria:

- Throughput [T]: number of input tuples being processed per time unit,
- Latency [L]: including both network and processing latency, which measures the time it takes for a data item to get ingested into the executed physical workflow and get out of it as a processed tuple,
- Communication Cost [C]: which quantifies the amount of communicated data throughout the network as an indicator of the bandwidth consumption incurred by a chosen physical plan,
- Energy Consumption [E]: this is an optional criterion included to measure the total amount of energy getting consumed by the running physical workflow(s). It is often useful to measure the lifetime of the network (recall that the network also consists of battery powered devices and loses its functionality as more devices become non-operational due to energy drains).

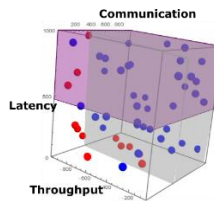


Figure 38: Illustration of Pareto Optimal Physical Plans

CREXDATA optimized analytics-as-a-service algorithms seek to find **Pareto optimal** solutions (physical plans). That is, physical plans that are not dominated, in each and every of the aforementioned performance dimensions, from others. Figure 38 provides an illustration of how a number of possible physical execution plans can be represented in a three-dimensional space encompassing the first three of the aforementioned metrics. Blue dots correspond to physical plans/workflows that are dominated in every dimension by others. Pareto optimal physical plans are those whose performance corresponds to the red dots. In order to choose among the Pareto optimal physical plans (red dots), CREXDATA optimization uses a convex combination of the various performance [T,L,C] measures.

Each physical workflow is represented (as also shown in Figure 37) as a graph, where each operator is mapped to its physical execution specifications. We call the initial workflow devised at time t_0 in the example of Section 5.1 as the root plan. But, in general, we term the

currently executed physical plan (at any time t_k) as **the root plan**. While the root plan is executed, the CREXDATA optimization algorithm keeps operating at the background and explores more, potentially better physical execution plan opportunities. To do that, each of the algorithms in our algorithmic suite operates on a **Graph of Graphs (GoG)**.

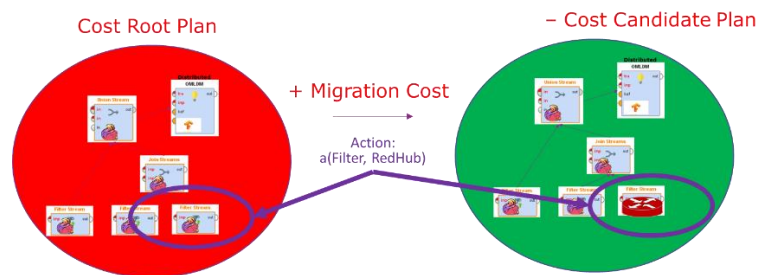




Figure 39: Two Nodes of the Graph of Graphs representation of the CREXDATA streaming analysis as a service

Each node of the GoG includes a physical workflow (i.e., a nested graph). Note that in the general case a node of the GoG may include a set of nested graphs if we want to simultaneously optimize many logical workflows. Nonetheless, for ease of exposition we henceforth assume a single logical workflow is being optimized and thus, a nested graph is a possible physical workflow of a single logical one.

An edge between nodes of the GoG is an **action** that performs a single change to the execution options or site(s) of just one operator of a physical workflow. As an example, Figure 39 shows a root plan with all operators placed at the cloud side. This plan is the red node of the GoG. To visit/explore the performance of another, green node of the GoG, an action that migrates the rightmost, bottom operator of the root physical workflow, from  to , is applied. This action yields a new physical plan with its own overall cost and a migration cost. The migration cost involves the cost that should be accounted for increased latency or zero throughput during if we decide to switch between the root and the candidate new plan. If $\text{Cost Root Plan} - \text{Cost Candidate Plan} + \text{Migration Cost}$ is a very negative (or very positive if we reverse the signs) value, this means that, for instance the green plan in Figure 39, improves per capita the root plan. Therefore, the green physical plan should be deployed instead.

But, of course, the green plan is just one node of the GoG, visited based on a single action. There are other actions in the root plan that can be applied in order to lead to other green nodes, i.e., **candidate physical plans**. Moreover, upon applying two actions instead of one, we have candidate physical plans that are two hop neighbours of the root plan. Two-hop neighbours are physical plan yielded from the root one after applying 2 actions (changes), three-hop neighbours are yielded from applying 3 actions to the root plan and so on. Therefore, the search space over which the CREXDATA algorithmic suit operates looks as depicted in **Figure 40**.

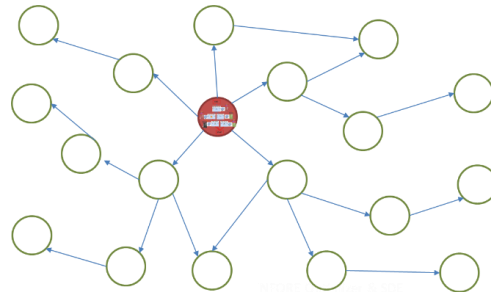


Figure 40: Graph of graphs (GoG) as the search space encompassing all possible physical plans - workflows for a given set of workflows and a network setting. Each node in the figure includes a nested graph which is a physical plan/workflow. Edges correspond to simple, single actions (changes) leading from one physical workflow to another.

5.3 Algorithmic Suite

5.3.1 Exhaustive Search (ESC) Algorithm

CREXDATA's exhaustive search approach creates the physical plan space by enumerating all possible nodes of the GoG using counting and number base switch. The algorithm is based on the enumeration of all possible plans using **workflow signatures**. Workflow signatures are formed as follows. For O operators, equivalently termed vertices in the scope of our algorithms, P execution options per operator/vertex and S sites, the possible plans are: $(P * S)^O$. We consider each vertex as a vector that can take $P*S$ values and construct an index into the O vectors as a O -digit, base- $(P*S)$ number. Each digit of this number is an index into one of series of vertex vectors representing a nested physical plan in the GoG.

For example, for a flow with 4 vertices and 4 possible values (e.g., 2 execution options (as an example, replication 2 or no replication) and 2 "change site" actions), we count from 0 to $4^4=256$, convert each number to 4 base-4 digits, and use these digits as indices into the vertex vectors to get a physical plan which corresponds to a nested node in the GoG. In this example, the possible actions are $[P1|S1, P1|S2, P2|S1, P2|S2]$. Hence, the number 164 maps to 2210 after base change, which represents the plan: $O1(P2S1) \rightarrow O2(P2S1) \rightarrow O3(P1S2) \rightarrow O4(P1S1)$. The pseudocode of the parallel Exhaustive Search with Counting (ESC) is detailed in Algorithm 2.

ESC is straightforwardly parallelizable as each worker can compute a single plan and its cost independently from the other workers. The algorithm explores the entire search space of physical plans.

In Lines 4-5 of Algorithm 2, the algorithm starts with the root plan and its cost. Line 6 starts a pool of threads for parallelization purposes and Line 7 computes the cardinality of the search space (number of possible physical plans). The `for` loop in Lines 8-11 takes in the sequential number of a plan and creates tasks for the workers. Each task that is created with the purpose of executing an instance of the `planWorker` method receiving as parameters the root workflow, the list of possible actions that may be applied to it and the plan counter i . As noted in Line 10, each such task is executed as soon as a worker becomes available.

The job of the `planWorker` method in Line 17 is to translate the plan number to an actual plan signature using the change base and the translation process into vertex vectors as

previously described in this section. Then, having the root workflow and the i -th candidate physical workflow, the function initially distinguishes which actions need to be applied in order to make the transition from the root to the i -th plan (Lines 21-23). Having done that, Lines 25-28 compute the plan cost and update, if needed, the global best plan and the accompanying, minimum so far, cost.

```

1  Algorithm: ESC
2  Input: <Graph> workflow, <List> actions, <int> k
3  Output: Optimal plan
4  minPlan ← workflow;
5  minCost ← workflow.cost;

6  start a pool WP with k workers;
7  #plans ← (#platforms * #sites) ^ #vertices;
8  for (int i; i < #plans; i++) {
9      w ← planWorker(workflow, actions, i);
10     WP.execute(w);    // runs once there is an available worker
11 }

12 terminate WP;
13 return minPlan;

14 Algorithm: planWorker
15 Input: <Graph> workflow, <List> actions, <int> i
16 Output: updates global variables minPlan, minCost
17 action[ ] n ← map i into actions; // n contains vertex placements ordered by vertex position
18 g ← workflow                      // work on a copy of flow
19 int c ← 0;
20 for <Vertex> v in g {
21     action a ← n[c++];           // get action at position c, which corresponds to the c-st vertex in g
22     apply a on v;                 // e.g., change v placement (site/execution option)
23     update v dependencies;        // modify v instances in all v'.adjVertices for {v'∈g, s.t. V'≠v}
24     // compute new workflow cost (each action results into a new plan)
25     c ← g.computeCost();
26     if (c < minCost) {
27         minCost = c;
28         minPlan = g;
29     }
30 }

```

Algorithm 2: CREXDATA Exhaustive Search Algorithm Pseudocode

5.3.2 Greedy Search (GSP) Algorithm

The Greedy Search Algorithm (GSP) of CREXDATA's algorithmic suite operates in steps (hops starting from the root plan on the GoG) aiming at finding progressive global optima at each step. In other words, GSP creates the GoG by following a greedy plan enumeration with progressive global optima. At each step, the algorithm progresses with the best (i.e., cheapest) single action across the entire workflow and repeats until all assignments are completed. The process is as follows:

1. Start with a workflow (initially the root one)
2. Find the action (one hop neighbour) that produces the lowest physical workflow cost
3. Update the workflow graph with the action
4. Repeat starting from the best plan found so far

The process is detailed in **Algorithm** . For better illustration we will describe the rationale of the algorithm using its visual representation in **Figure 41**.

```
1  Algorithm: GSP
2  Input: <Graph> workflow, <List> actions
3  Output: A plan with the minimum cost found
4  g ← workflow;                // work on a copy of flow
5  g' ← null;                   // a graph to keep an intermediate state
6  minV, minA ← null;           //to keep vertices with minimum cost action and the action
7  <List> vlst ← all <Vertex> v of g;
8  while (vlst is not empty) {
9    minCost ← ∞;
10   g' ← g;
11   // find the cheapest action
12   for vertex v in vlst {      // for each vertex
13     for action a in actions {  //try all actions
14       apply a on v;           // e.g., change v placement (site/execution option)
15       update v dependencies;   // modify v instances in all v'.adjVertices for {v'∈g',s.t. V'≠v}
16       c ← g.computeCost();     // compute new workflow cost
17       if (c < minCost) {       // keeps the cheapest action and the vertex with that action
18         minCost ← c;
19         minV ← v;
20         minA ← a;
21       }
22     }
23   }
24   // update intermediate state
25   g ← g';
26   apply minA on minV;
27   update minV dependencies;
28   g.computeCost();
29   // remove the visited vertex and repeat
   vlst.remove(minV);
```

```

30 minV, minA ← null;
31 }
32 return g;
33

```

Algorithm 3: CREXDATA Greedy Search Algorithm Pseudocode

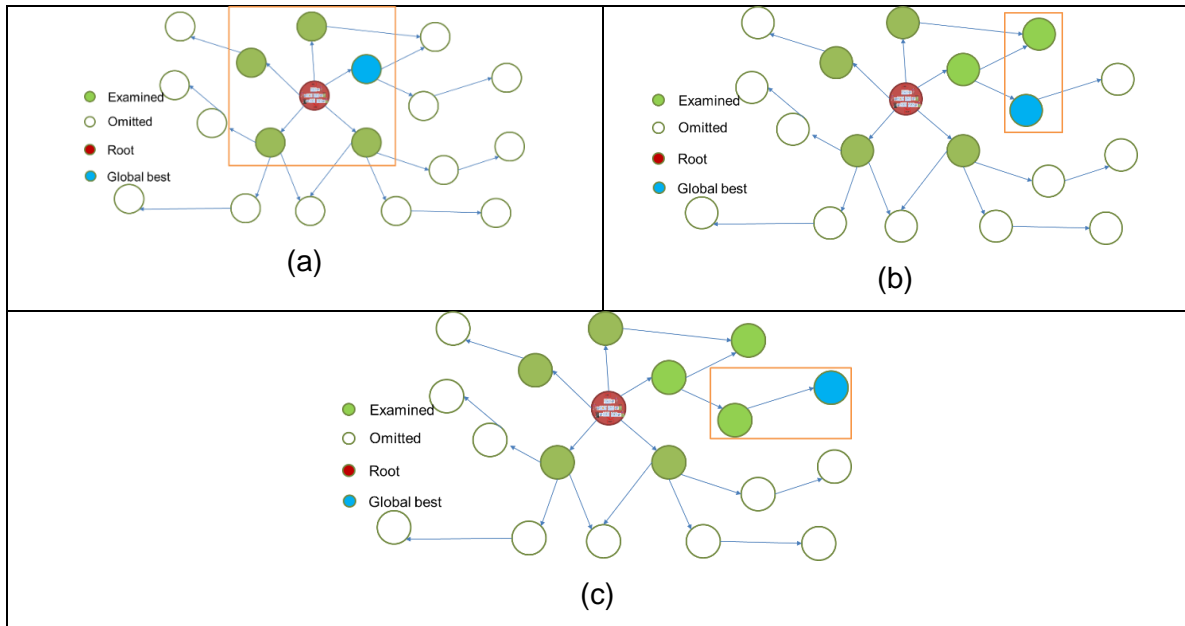


Figure 41: Illustration of the Greedy Search Algorithm Operation

Figure 41 shows 3 iterations of the GSP algorithm. In Figure 41a, the algorithm starts from the red, root plan and computes the costs of the one hop neighbours, yielded after applying possible actions. Out of the examined plans, the light blue-coloured one is the current global optimum. Therefore, the algorithm uses this plan and repeats the process by examining the actions and computing the costs of its one hop neighbours. In Figure 41b, out of the two possible actions, the newly light blue-coloured one is the new global optimum. In the third iteration of the algorithm, the one hop neighbours of the latter plan are examined. In Figure 41c we observe that only a single one hop neighbour exists and gives a new global optimum and so on.

The greedy solution is suboptimal, as the decision at each step does not consider the long-term value of its action, but it is extremely fast requiring minimal execution time. This is important in highly volatile settings where network devices enter or leave frequently, as well as when stream statistical properties often change. If in such settings the optimization algorithm needs a considerable amount of time to return a preferable physical plan, its suggestion may be outdated by the time it is returned. GSP's speed gracefully handles such situations. GSP can be parallelized either at the vertex or the action level, but our experimental results show that this is not necessary as its execution time is minimal even for large scale networks.

5.3.3 Random Sampling Search (RSS) Algorithm

The RSS algorithm computes a random sample of the possible physical plans and returns the best solution amongst them. The algorithm builds on top of the ESC technique. Recall that ESC creates the plan space by enumerating all possible states using counting and number base switch. Instead of enumerating all possible plans (and thus finding an optimal solution), RSS instantiates a sample of them. An exemplary illustration is provided in Figure 42.

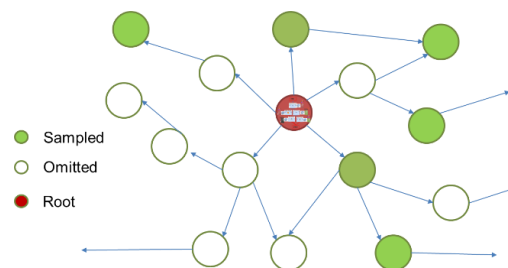


Figure 42: Illustration of the Random Sampling Search Algorithm Operation

Algorithm 4 outlines RSS operation. RSS uses the same `planWorker` method as ESC does (Line 15), but it instantiates it only for the sampled physical plans (Lines 11-13). The solution returned is suboptimal, as the random selection of plans does not guarantee finding the optimal plan.

```

1  Algorithm: RSS
2  Input: <Graph> flow, <List> actions, int sampleSize
3  Output: A plan with the minimum cost found
4  minPlan ← flow;
5  minCost ← flow.cost;
6
7  #plans ← (#platforms * #sites) ^ #vertices;
8
9  if (sampleSize >= #plans) sampleSize = #plans;    // if the plan space is small do ESC
10
11 <List> sample ← get sampleSize unique numbers in [0,#plans);
12
13 for int i in sample {
14     // work similarly to ESC
15     planWorker(flow, actions, i);
16 }

```

Algorithm 4: CREXDATA Random Search Sampling Algorithm Pseudocode

The advantage of RSS over GSP is that it cannot be trapped to local optima (global optima per step as computed by GSP). Furthermore, the execution time of the algorithm is in direct relation to the chosen sample size. Therefore, depending on the volatility of the optimization setup may be a more preferable choice compared to GSP. Our experimental evaluation in

Section 5.5 provides a practical comparison of ESC, GSP, RSS in practical workflows and small, medium, large-scale networks of devices.

Since CREXDATA may, in some cases, also need to process static data stored at a database (for instance terrain and other information in the Weather Emergency Use case or coastal area information in the Maritime Use Case) using used defined functions (UDFs) expressed as a RM Studio operator, we handled such cases separately. More precisely, we designed QFusor, an optimizer plugin for UDF queries over relational databases. QFusor minimizes the performance overheads introduced upon using UDFs in SQL execution environments by employing techniques such as vectorization, parallelization, tracing JIT compilation, and operator fusion for various, commonly used and supported in RapidMiner Studio, types of UDF also engaging relational operators. QFusor is engine-agnostic and can work with several popular SQL databases some of which are supported by RapidMiner Studio in the overall CREXDATA architecture.

5.4 Statistics Collection and the Need for Simulators

For our algorithms to be useful in devising a physical plan of good practical performance, they require accurate statistics. These statistics are to be used in the `computeCost` function they all incorporate. In the past, various approaches have adopted System-R like techniques or machine learning models [100] [101] to incorporate statistics and use them to measure the overall performance of physical workflows. Nevertheless, the setups over which CREXDATA operates are different. The volatility of the network as well as the arbitrary size of workflows, networks, execution options and the variety of possible operators, together with the combinatorial optimization space, do not leave room for making the assumption that there exist neither accurate analytic formulas nor tractably trainable machine learning models that can capture the behaviour of combinations of operator physical implementations. Therefore, in CREXDATA we resort to derive statistics via truthful simulators possibly complemented with runtime statistics of the physical plans that are actually deployed.

Nonetheless, this is easier said than done. We surveyed existing simulators, based on desired features including (a) number of citations and number of Github stars, (b) the programmability of the simulator (the ability to actually code operators and workflows), (c) the level of the cloud to edge continuum that can be covered by the simulator and (d) the coverage it can provide, directly or via derived measures, to the performance criteria in Section 5.2. The result of our survey is summarized in **Table 7**.

Table 7: Candidate simulators and their adoption/popularity, metrics collection, cloud to edge continuum coverage features

Simulator	Github */ Paper Citations	Program - mable	IoT Layers Covered	Metrics
iFogSim [102]	175 / 1610	X	Edge, Fog Cloud	Execution time per tuple, Network usage, Energy consumption, Cost of infrastructure
EdgeCloudSim [103]	393 / 539	X	Fog, Cloud	Latency (per task)

Simulator	Github */ Paper Citations	Program - mable	IoT Layers Covered	Metrics
PureEdgeSim [104]	149 / 40	X	Fog, Cloud	Latency (per task)
IoTNetSim [105]	0 / 28	X	Edge, Fog Cloud	-
IoTsim-Edge [106]	13 / 120	X	Edge, Fog	Energy consumption, latency(per event)
IoTsim-Stream [107]	3 / 17	X	Cloud (Multicloud)	CPU, RAM usage
IoTsim [108]	0 / 255	X	Edge, Cloud	VM cost, execution time
Raspberry Pi emulation ¹³	-	Y	Edge	-
Raspberry Pi cluster	-	Y	Fog	Custom metrics via Prometheus

Based on **Table 7**, we resort to iFogSim (version 2) due to its popularity, layer coverage and richness of performance metrics. Nevertheless, iFogSim is not directly usable for simulating any valid physical plan in the scope of CREXDATA. Its limitations can be summarized as follows: (i) communication links are not bidirectional, therefore if one operator is assigned to a device in the fog, it cannot provide its output if the upstream operator is in the cloud, (ii) it cannot automatically receive a physical workflow from the rest of the CREXDATA architecture and deploy/ simulate it over a chosen network (iii) it does not provide all the target performance measures i.e., end to end latency and throughput as well as state size for migration cost calculation purposes are missing.

To tackle the above limitations, we significantly redesign iFogSim. As illustrated in **Figure 43**, iFogSim is now only a small sub-system of the CREXDATA Simulator Executor and Statistics Collector. Our contributions are the two sub-systems surrounding iFogSim, namely the workflow-placement-generator and ifogsim-wrapper. The workflow placement generator is able to generate topologies and workflows according to the specifications of the CREXDATA architecture. And then generate simulations for these workflows using the placement generator interface. Here, we use the strategy software pattern to have different algorithms for the placement generation. We plan to provide CREXDATA Simulator Executor and Statistics Collector open-source as soon as this effort is completed.

¹³ M. Stawiski, "Emulating a raspberry pi in qemu," <https://interrupt.memfault.com/blog/emulating-raspberry-pi-in-qemu> , June 2023.

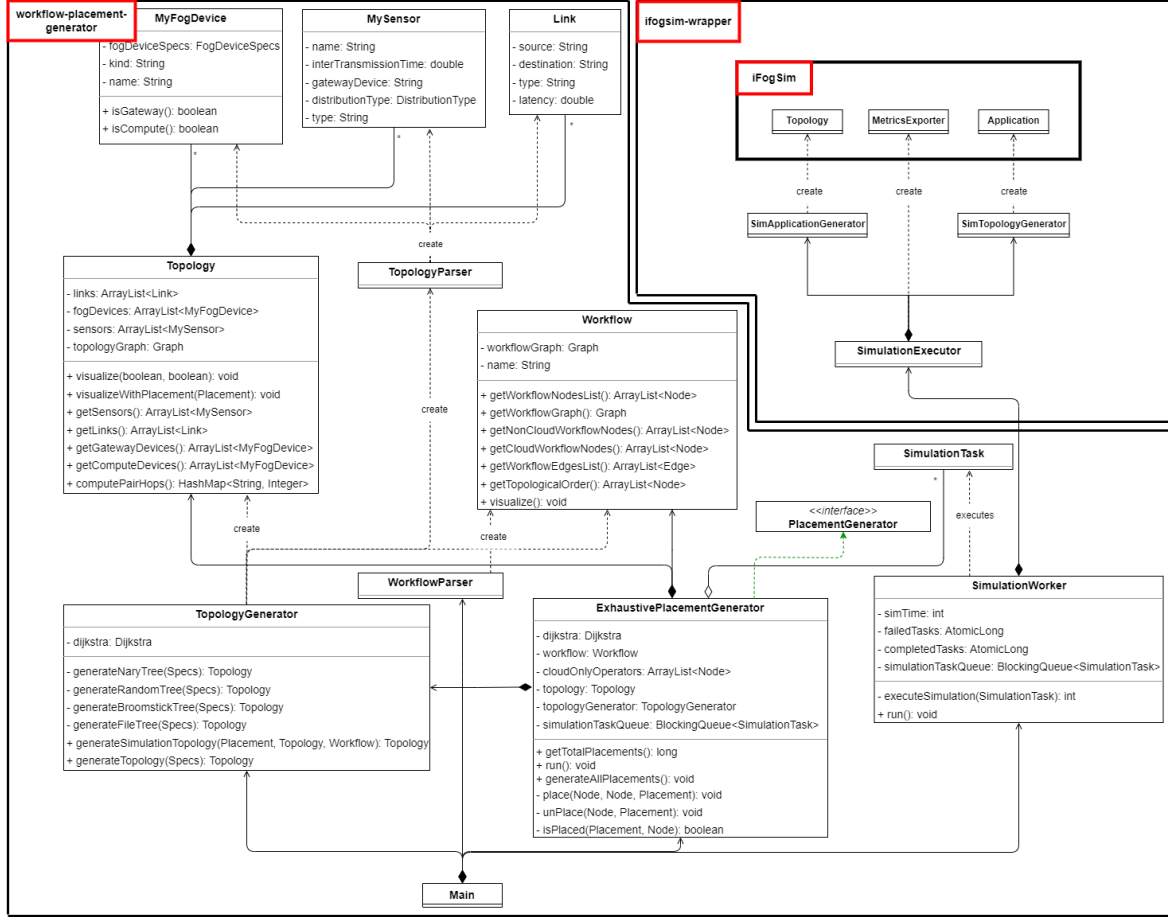


Figure 43: CREXDATA Simulator Executor and Statistics Collector significantly extending iFogSim

Finally, because, in principle, there are categories of operators that are only deployable on the cloud, such as heavy-duty machine or deep learning operators and these operators introduce parallelization and CPU-GPU related execution trade-offs, we studied the statistical features and statistical properties that affect these categories of operators separately from the core operators of CREXDATA. In particular in [97] we statistically analyse the key factors affecting the execution performance of task(operator)-based workflows on a High Performance Computing (HPC) infrastructures composed of heterogeneous CPU-GPU clusters. Several results on interrelated factors regarding the execution options (physical execution implementation) of the task algorithm, dataset, devoted resources, and system utilization employed are revealed. We consider this statistics collection and analysis methodology as the first step towards an automated method to optimize task-based workflows in modern, high-compute capacity, CPU-GPU engines.

5.5 Experimentation

We measure the performance of ESC, GSP and RSS algorithms over a wide variety of network settings, from small (10s of sites) to medium (100s of devices) and large (1000s of devices) scale networks, using the TRAIN workflow from the well-known, highly cited RIoT

benchmark [109], illustrated in **Figure 44**. Please refer to [109] for further details on the utilized operators. Our experiments were run on a server with (a) 2 Intel Xeon Silver 4310 processors with 12 cores and 24 threads each, (b) Four 64GB RAMs RDIMM 3200MT/s each, (c) One ROM 960GB SSD vSAS with read-intensive 12Gbps.

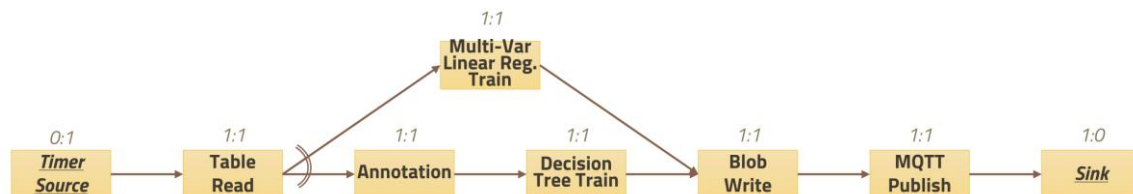


Figure 44: TRAIN workflow used in our experimental evaluation¹⁴

Figure 45a and **Figure 45b** provide a comparative analysis of the involved algorithms regarding their execution time and the goodness of the output physical plan, respectively. ESC and RSS were capped at 100 and 10 seconds, correspondingly. This means that if they have not output a physical plan by examining the entire search space by 100 or 10 seconds, respectively, they are enforced to output the best plan found so far. GSP has no such restriction since its execution time is always in the order of a few seconds. For reasons explained in **Figure 45c**, the parallelism of ESC is set to 10. Remarkably, RSS and GSP currently run with a parallelism of 1 since we want to also judge whether they are worthwhile to parallelize based on their execution time and goodness of provided solutions. As **Figure 45a** and **Figure 45b** demonstrate, for an order of magnitude reduction in execution time (either due to the cap or without it) in **Figure 45a**, RSS approaches the goodness of ESC output physical plan, providing near optimal solutions. On the other hand, GSP is two to three orders of magnitude faster than RSS and GSP, correspondingly, but with a considerably suboptimal solution each time.

Turning our attention to **Figure 45c**, we observe that the parallelism of ESC reaches the maximum number of examined physical plans and, thus, the portion of the entire search space that is inspected by the algorithm, when the parallelism is set to 10. This is because for more threads the benefits of examining multiple physical plans in parallel is outweighed by computational, thread synchronization barriers and locks on atomic variables. Therefore, in **Figure 45a,b** parallelism is set to 10.

Another subtle algorithmic parameter for ESC is the size of queue size that will be used for enqueueing candidate physical plans for the various workers to dequeue and examine their performance. We experimented with different queue sizes ranging between 100*parallelism, 1000*parallelism to 10000*parallelism. According to the plot of **Figure 45d**, the lowest execution time is incurred for queue size of 1000*parallelism. This is because for 100*parallelism the queue is often left empty while, for 10000*parallelism, the execution time is negatively affected by memory to CPU communication lags.

¹⁴ <https://github.com/dream-lab/riot-bench>

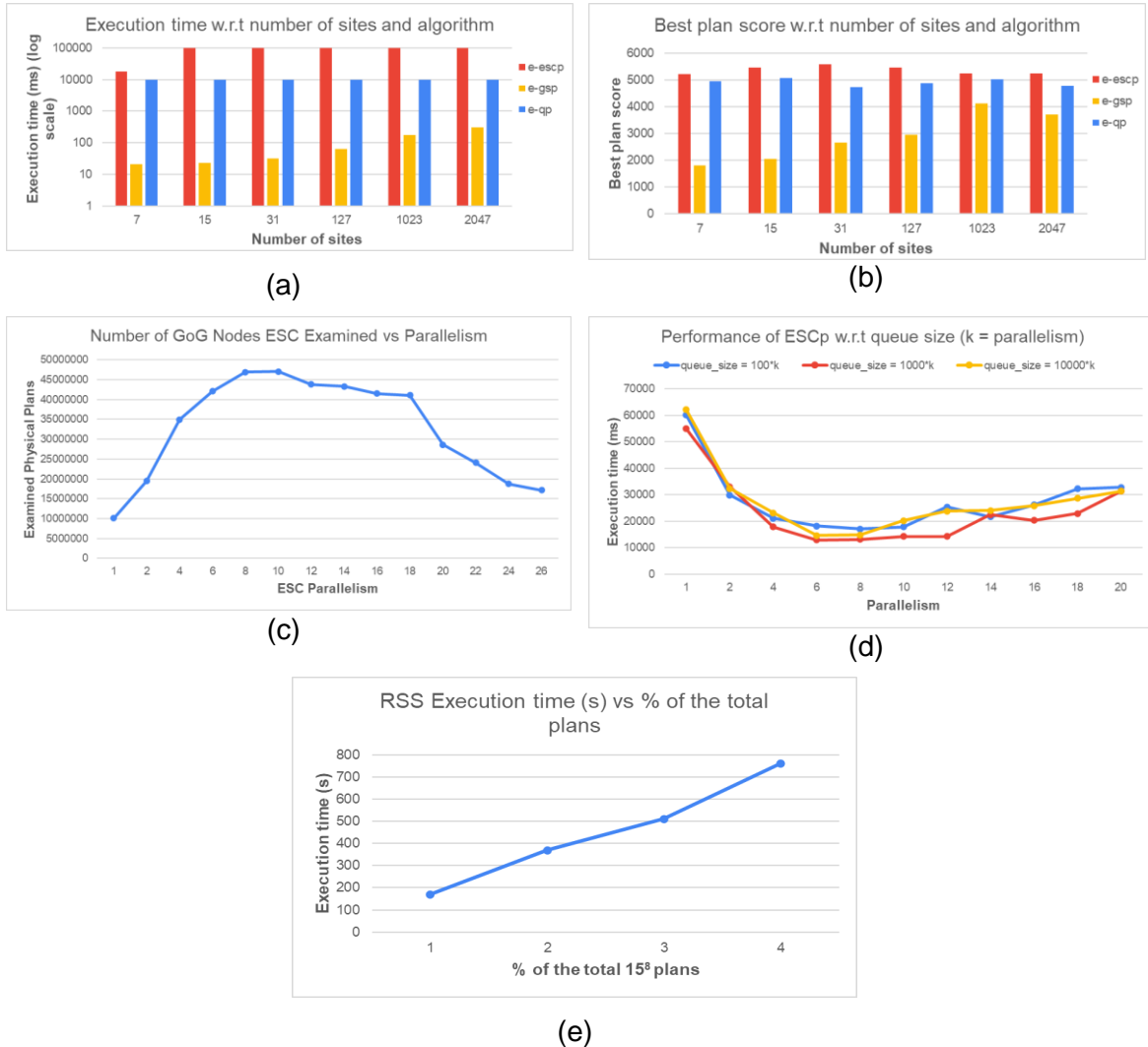


Figure 45: Scalability Analysis and Parameter Tuning for the CREXDATA Algorithmic Suite (ESC, RSS, GSP Algorithms)

Finally, we examine the potential benefit of parallelism for the RSS and GSP algorithms, which are currently both single threaded. As shown in **Figure 45a**, the execution time of GSP is not severely affected by increasing the network size. In the figure, we have an order of magnitude increase in execution time for three orders of magnitude increase in network size for GSP. On the other hand, **Figure 45e** shows the execution time of RSS for sample sizes of 1-4% for a search space composed of 15^8 physical plans (for a network of 15 devices and the workflow of Figure 44 which is composed of 8 operators). As plotted in the graph, the execution time of RSS exhibits a linear trend with the increase of the sampled search space. Consequently, parallelization plays an important role in the execution time of the algorithm over networks composed of 100s or 1000s of devices. Additionally, we expect that GSP will be worthwhile to parallelize for some workflows within the scope of CREXDATA which will be expectedly composed of more operators and higher branching factor in its operators.

6 Text mining for Event Extraction

In this section we describe the text mining modules developed within the context of Task 4.5. In essence, we are describing the algorithms and language models behind these modules which have been developed to create a social media toolkit for real-time information extraction from multilingual text messages on social networks to support civil protection authorities in the face of weather-induced emergencies.

Decision making and planning during weather emergency situations require authorities to study environmental conditions from different sources. This could include meteorological data, weather nowcasts, environmental sensors such as hydrological and forest fire emission sensors, and autonomous devices (e.g. drones or robots). While these sources offer valuable perspectives, obtaining real-time information from individuals directly involved in the emergency can improve response strategies. This information can be obtained from geolocated-social media posts published by local government agencies, first responders, bystanders, and affected persons sharing updates and alerts on the developing event.

Acquiring and mining useful information about emerging events from social media poses several challenges, including volume of data, unstructured and ungrammatical (internet jargon) posts, and multilinguality. Large language models (LLMs) are a suitable solution to these challenges. Pretrained transformer-based multilingual models that utilise attention mechanisms can be fine-tuned to capture the nuances of the language used in the social media ecosystem. By leveraging these models, we can effectively identify informative social media posts. Subsequently, key information can be extracted from these posts for further analysis.

The results in this section directly contribute towards the main objective (MO1.3) of CREXDATA. Specifically, in developing a multilingual large language model (MLLM) that will enable end users to monitor social networks for information on weather emergencies. To achieve this, a Bidirectional Encoder Representations from Transformers (BERT) model is fine-tuned using social media posts from real weather emergencies crawled mainly from X (Twitter) to identify relevant posts in connection with the emergency, also detecting and classifying the type of the event currently unfolding, specifically wildfires, and floods. The resulting model is equipped with an interface to a streaming pipeline built with Apache Kafka, to process social media posts in real-time after a trigger condition is met. Relevant posts are then served to a LLM-based question-answering (QA) system specifically designed to extract information from disaster-related text data. By enabling users to formulate focused queries regarding the unfolding event, the system aims to extract valuable insights.

In the following subsections, we provide detailed descriptions of the BERT model used for event type prediction and the question answering model used for information extraction.

6.1 BERT Based Event Type Classifier

As discussed previously, the main goal is to monitor social media networks to identify, track and provide meaningful information from ongoing disasters, specifically fires and floods. To acquire social media posts the system needs access to the Application User Interface (API) of the end user's preferred social network. This social network should ideally be a microblogging platform with a focus on textual posts like Twitter, Mastodon, etc. Most of

these platforms no longer provide free API access, but rather paid tiers based on the frequency and number of downloaded posts. In choosing a social network to monitor, it is also important to study the population of the interested area to determine their preferred networks. For example, Twitter might be preferred in North America, while Facebook is preferred in Germany. For experimentation and proof of concept, social media posts about past weather emergencies from Twitter will be used to simulate real time post streaming from a social networking site.

The first step of the system is to identify relevant posts. Relevance detection has been researched in previous works [110], but the relevance of a text is defined differently depending on the application. In the context of this task, a relevant post is one in which the person is speaking about an ongoing incident and providing information that can be used to assess the status of the event. On the other hand, posts that are not relevant may include sympathy, donation efforts, political discussions, or even conspiracy theories. After relevance detection, the next step is to classify the event type, which consists in analysing what type of incident the post is referring to (e.g. flood, fire). We explored the combination of both steps, into one text classification model which classifies posts as referring to a flood, fire, and none, where “none” refers to posts that are not relevant.

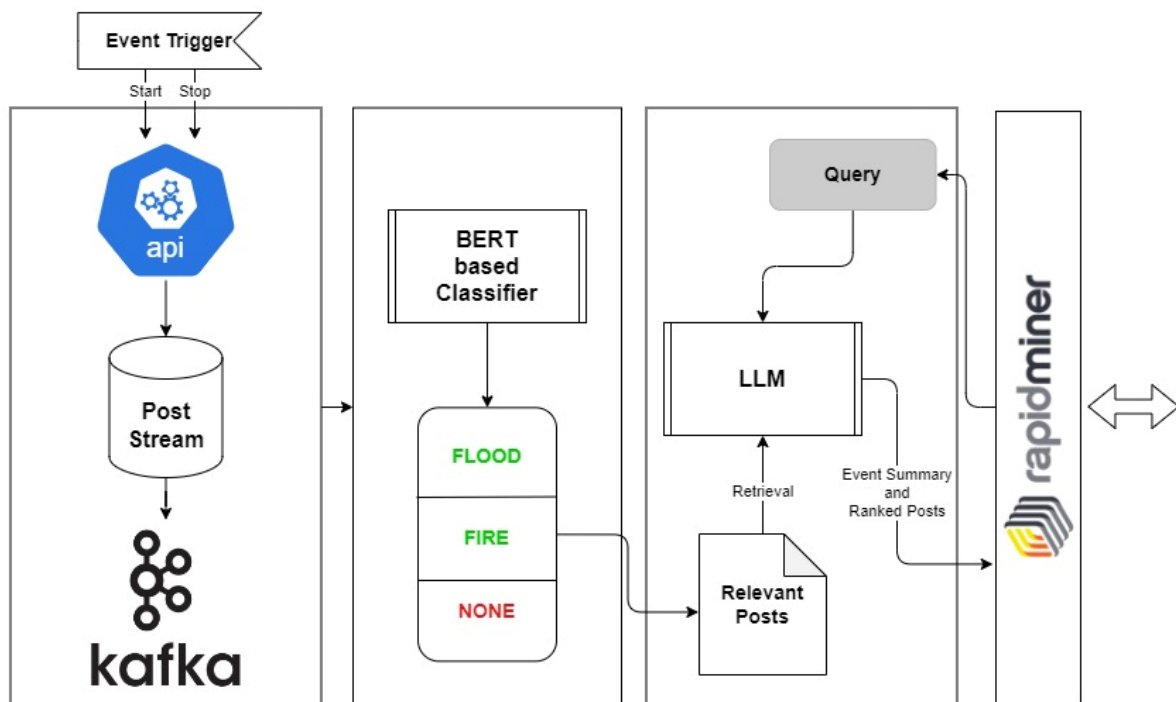


Figure 46: Architecture

6.1.1 System architecture

In **Figure 46**, we show the overall system architecture. Once a trigger condition for a specific weather emergency is met, social media posts are collected from affected geographic areas. The streams of posts collected from those areas are then classified by event type using our fine-tuned event type classifier. The collected relevant posts may then be queried by the

question answering model in order to generate a summary of the event and retrieve posts that answer specific questions for the end user.

Model

Our first approach to the event classification process was to have two models: one that predicted relevancy and another that predicted event type. The idea was to use a lightweight model to quickly discard the non-relevant posts (the majority) and one bigger model that receives relevant posts and classifies them according to which event they are referring to. After some experimentation, we discovered that we could use the same lightweight model to select relevant posts and classify event types simultaneously. Combining both tasks into one model is more time-efficient and produces good results.

The architecture of this model is based on mBERT, which is a BERT model trained using multilingual general-purpose text. BERT is a language model with ~100 million parameters that has been trained to predict missing tokens and on next sentence prediction [111]. It was introduced in 2018 and improved the state of the art in many natural language-based tasks. This pretrained model is usually further fine-tuned with task-specific data for other tasks such as text classification. This fine-tuning usually consists of training both the pretrained model and an additional classification head.

Train Data

The challenge with training text classification models is building an annotated dataset for the specific task, which is even more challenging when tackling multilinguality. We aim to cover as many languages as possible, although the number of languages supported is dependent on acquiring data related to fires and floods in the target languages. The currently focused languages are English, German, Spanish and Catalan. Our initial training dataset includes publicly available datasets from previous research on weather disasters detailed in D2.1. These datasets were mostly annotated for relevance to an event but not for event type classification. We convert this annotation by considering all relevant posts under a particular event as the type of that event (flood, or fire). Since most of these datasets are in English, we required more multilingual data. The following are some strategies we explored to acquire more data for the other languages.

Data Generation

We explored generating synthetic datasets by prompting generative LLMs. This solution has been explored for different purposes including training neural machine translation models. In our experiments, we used Mistral AI's 7 billion parameter model [112]. The model was provided with common keywords related to weather emergencies, and prompted to generate texts that resemble social media posts talking about those emergencies. We also tested providing some few shot post samples to the model. The resulting generated posts were very repetitive and lacking in ingenuity. We decided not to pursue generation as it could bias our model and reduce accuracy when combined with real data.

Data auto-annotation

Like in data generation, we explored the use of a generative LLM for zero shot text classification to auto-annotate social media posts for fine tuning. It has been shown that pretrained LLMs that have been instruction tuned (i.e. fine-tuned to respond to NLP instructions) perform reasonably for zero shot text classification which makes them useful for automatic data annotation [113]. We experimented using Mistral AI's 7 billion parameter

model for this purpose. The annotations produced by mistral 7B were comparable to annotations made by human annotators with above 80% accuracy. This method will be used to annotate more languages instead of manual annotations when necessary.

Data Translation

A common practice for dealing with data scarcity in multilingual tasks is generating synthetic data by using machine translation on an existing corpus in a different language. Datasets in a source language, most commonly in English, can be translated using neural machine translation (NMT) models into the other target languages. We have used this method to create synthetic datasets in German, Spanish, and Catalan. The data we had in English was translated to German¹⁵, Spanish¹⁶ and Catalan¹⁷ using state of the art open-source machine translation models. Because these datasets are not real-world datasets, we are manually annotating test datasets to evaluate the performance of models fine-tuned on the combination of real and synthetic data.

Fine-tuning

In state-of-the-art language modelling for natural language processing, models are pretrained on large datasets of general-purpose text, such as newspaper articles, Wikipedia articles, etc. which allows them to learn the semantics of natural language and embed general knowledge. This pretraining is very computationally expensive and time consuming. Once a model has been pretrained, it can be further trained (fine-tuning) for a specific NLP task by using smaller task specific datasets.

In text classification, the common fine-tuning process usually involves training the pretrained model and the classification head simultaneously. This is the fine-tuning technique we are reporting on in the results section. There are other approaches that aim at reducing training time, such as freezing the layers of the pretrained model and just fine-tuning the classification head, or more sophisticated approaches such as LoRA [14], which aims to fine-tune a small set of the total parameters while not losing too much performance in the process. We explored layer freezing and LoRA and found LoRA to produce slightly worse results than common finetuning but with a 16% decrease in training time, while layer freezing produce significantly worse results without much decrease in training time in comparison to LoRA as shown in **Table 8**. All models were fine-tuned on high performance computing nodes with 4x NVIDIA Hopper H100 64GB GPUs and 2x Intel Xeon Platinum 8460Y 2.3GHz CPUs.

¹⁵ <https://huggingface.co/Helsinki-NLP/opus-mt-en-de>

¹⁶ <https://huggingface.co/Helsinki-NLP/opus-mt-en-es>

¹⁷ <https://huggingface.co/projecte-aina/aina-translator-ca-en>

Table 8: Finetuning methods comparisons. The best performing result is indicated in bold. The second best is underlined for comparison

Architecture	Precision			Recall			F1			Accuracy	Training time (minutes)
	Fire	Flood	None	Fire	Flood	None	Fire	Flood	None		
mBERT (common)	<u>0.947</u>	0.956	0.99	<u>0.958</u>	0.96	0.976	<u>0.953</u>	0.958	0.983	0.965	16.73
mBERT (LoRA)	0.968	<u>0.939</u>	<u>0.929</u>	0.965	<u>0.938</u>	<u>0.933</u>	0.966	<u>0.938</u>	<u>0.931</u>	<u>0.946</u>	<u>14.03</u>
mBERT (frozen)	0.539	0.658	0.66	0.819	0.282	0.703	0.65	0.395	0.681	0.597	13.85

Test Data

A percentage of our dataset which includes public datasets (mostly in English), and synthetic datasets (translated data) was reserved for testing and evaluation purposes. This test dataset was composed of about 30% of the dataset and was shuffled during every training process. It also consisted of mostly real datasets. Although this practice has yielded good results in the event type classification, we are working on human annotation of real data that will be used solely for testing, which will produce more reliable results. To ease the process of manual annotation, we are using the current best iteration of our event type classifier, to detect posts that are not relevant to a weather emergency, then only manually annotating those which are relevant to the event.

6.1.2 Results

For experimentation, we considered mBERT¹⁸ (BERT architecture), XLM-R¹⁹ (Roberta architecture with CommonCrawl²⁰ dataset), and TwHIN²¹ (trained on social media messages) base models, and XLM-T²² fine-tuned on XLM-R base model with social media messages for sentiment analysis. Our focus is mBERT, but we include the other models because they alter the training process and/or modifying the source of the data used for training. For comparison, we use the precision, recall, f1 and accuracy metrics that are standard for text classification evaluation.

In **Table 9**, we made a comparison of all the base models without fine-tuning and mBERT fine-tuning for event type prediction. Naturally, the base models perform poorly because they haven't been trained for the task, with the TwHIN exception with an above average accuracy. This because TwHIN has been pretrained on social media messages and has gained knowledge on social media. The table also shows that the mean latency when using the models for classification is negligible even after fine-tuning. It is important to note that these

¹⁸ <https://huggingface.co/google-bert/bert-base-multilingual-cased>

¹⁹ https://huggingface.co/docs/transformers/en/model_doc/xlm-roberta

²⁰ <https://commoncrawl.org/>

²¹ <https://huggingface.co/Twitter/twhin-bert-base>

²² <https://huggingface.co/cardiffnlp/twitter-xlm-roberta-base-sentiment>

latency results are from a high-performance computing node with GPUs, as such they are better than they would be in lesser resource applications.

Table 9: Comparison of base models with a finetuned model. The best performing result is indicated in bold. The second best is underlined for comparison

Model name	Precision	Recall	F1	Accuracy	Mean latency
mBERT	0.3461	0.3323	0.0489	0.0534	<u>0.0135</u>
XLM-R	0.3900	0.3333	0.0969	0.1701	0.0133
XLM-T	0.3081	0.3239	0.0856	0.1157	0.0133
TwHIN	<u>0.4014</u>	<u>0.3739</u>	<u>0.3414</u>	<u>0.5488</u>	0.0155
mBERT (Fine-tuned)	0.8800	0.9651	0.9180	0.9504	0.0136

Next in **Table 10**, we show a comparison of the base models all fine-tuned with a dataset balanced according to classes (“flood”, “fire”, and “none”). The dataset is balanced to reduce bias to certain classes. The train set includes approximately 2500 social media messages in four different languages, German (DE), Spanish (ES), Catalan (CA), and English (EN), while the test set includes approximately 1200 messages in the same languages.

The fine-tuned based models all have above 90% performance in all language and class precision with negligible differences. The differences lie in the precision of the fire and flood classes between models pretrained with general-purpose text (mBERT, XLM-R) and those trained with Twitter data (XLM-T, TwHIN). The latter models show a precision increase for the fire and flood classes, resulting in a higher F1 score. Our hypothesis for these results is that XLM-T and TwHIN, having been pretrained with Twitter data, are more attuned to informal language and social media jargon.

In terms of language, the models show comparably a slightly better performance in English, probably because English being the dominating language on internet, these models are pretrained on more English samples than other languages.

6.2 Question Answering for Event Information Extraction

During disasters, a critical need arises for timely and accurate information. Emergency responders, policymakers, and the general public all rely heavily on this information to guide their actions. However, the event of a disaster often brings a vast amount of text data (e.g. news reports, social media updates, etc.) creating a complex information landscape. Sifting through this overwhelming volume of data to find specific, important details can be a challenging task, hindering effective response and recovery efforts.

To address this challenge, we propose a novel Question Answering (QA) method. Our system aims to extract crucial information from disaster-related text data by enabling users to ask focused questions about the unfolding event. By providing clear and concise answers,

the QA system could serve as a powerful tool to improve information access and communication during critical times, empowering informed decision-making.

Table 10: Comparison of base models fine-tuned for event type prediction

Architecture	Language	Precision			Recall			F1			Accuracy
		Fire	Flood	None	Fire	Flood	None	Fire	Flood	None	
mBERT	DE	0.955	0.942	0.99	0.941	0.966	0.977	0.948	0.954	0.983	0.962
	CA	0.967	0.934	0.971	0.938	0.959	0.975	0.952	0.946	0.973	0.957
	ES	0.964	0.929	0.981	0.928	0.97	0.977	0.945	0.949	0.979	0.958
	EN	0.947	0.956	0.99	0.958	0.96	0.976	0.953	0.958	0.983	0.965
XLM-R	DE	0.971	0.941	0.957	0.977	0.963	0.926	0.974	0.952	0.941	0.957
	CA	0.96	0.941	0.961	0.976	0.953	0.934	0.968	0.947	0.948	0.954
	ES	0.97	0.931	0.958	0.976	0.962	0.924	0.973	0.947	0.941	0.953
	EN	0.975	0.948	0.958	0.974	0.965	0.941	0.974	0.956	0.949	0.96
XLM-T	DE	0.965	0.961	0.958	0.992	0.953	0.936	0.978	0.957	0.947	0.962
	CA	0.947	0.969	0.959	0.988	0.941	0.947	0.967	0.955	0.953	0.958
	ES	0.963	0.953	0.956	0.986	0.954	0.935	0.974	0.953	0.946	0.958
	EN	0.971	0.965	0.948	0.981	0.953	0.95	0.976	0.959	0.949	0.962
TwHIN-BERT	DE	0.978	0.953	0.953	0.985	0.958	0.94	0.981	0.955	0.947	0.962
	CA	0.957	0.962	0.958	0.985	0.942	0.95	0.97	0.952	0.954	0.959
	ES	0.979	0.948	0.958	0.983	0.958	0.944	0.981	0.953	0.951	0.961
	EN	0.977	0.962	0.958	0.984	0.959	0.953	0.98	0.96	0.956	0.966

6.2.1 Methodology

With the rise of Large Language Models (LLMs), a great amount of research focused on how to leverage LLMs for information retrieval tasks, such as zero-shot retrieval. Early methods compared each query-document pair with a score, picking the highest-scoring ones. Researchers have improved these methods by adding more context, using LLMs to generate extra queries, summaries, or relevant details. This enrichment significantly boosted retrieval performance, especially for unseen (zero-shot) queries.

Typically, retrieval systems embed queries and documents in a shared space for efficient searching. While recent methods leverage LLMs to enrich retrieval in various ways, the ability to improve the results of the retriever, based on the re-ranking stage remains limited.

Additionally, LLMs can generate inaccurate content, and their performance can be affected by factors like prompt order.

To address these issues, some studies [115] [116] propose using LLMs as relevance assessors, providing individual assessments for each query-document pair. These approaches aim to enhance trustworthiness by leveraging the LLM's strengths in understanding nuances and identifying potentially irrelevant content.

However, most existing methods treat retrieval and relevance assessment as separate tasks, missing potential benefits from combining them. Our approach bridges this gap by merging individual rankings from separate retrieval and relevance assessment stages using rank aggregation techniques. This allows our method to exploit the strengths of both stages, leading to a more accurate final ranking of retrieved documents.

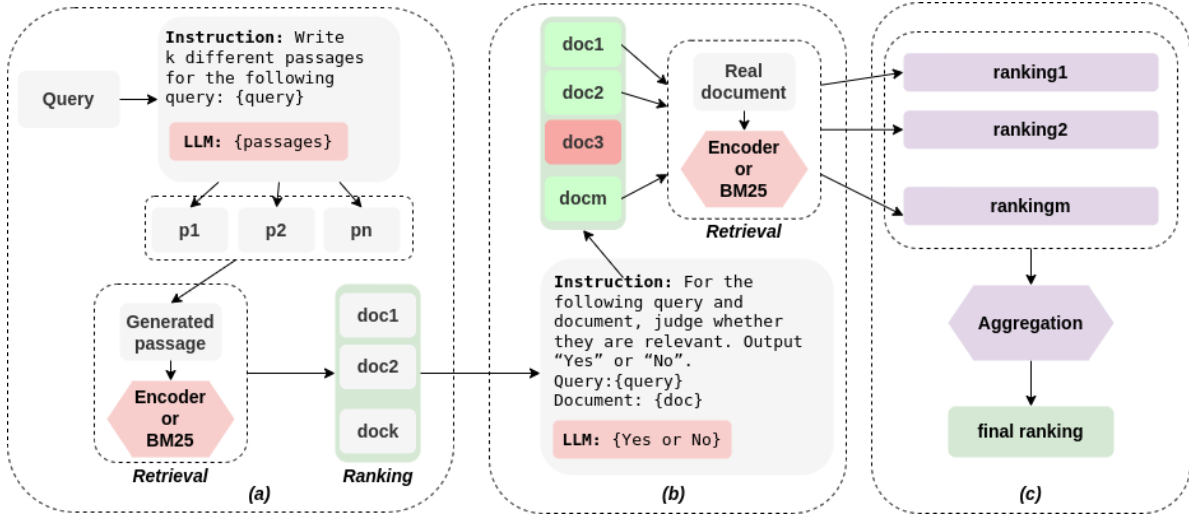


Figure 47 The system architecture of the QA model

System Architecture

In Figure 47, we present the proposed approach, which consists of three main steps. Initially, the retriever leverages LLM-generated query-relevant passages to identify candidate documents from the corpus. Subsequently, the retrieved documents undergo an LLM-based relevance assessment, where only documents deemed relevant by the LLM progress to the next stage. Then, for each document, a separate ranking is produced. Finally, a rank aggregation technique merges individual rankings leading to a more accurate final ranking. Notably, our approach relies on LLM inference for both retrieval and relevance assessment, without the need for separate training steps. Below, we describe each step in detail.

Passage Generation

Building upon prior work [117] that highlights the advantages of enriched query and document representations, our method focuses on expanding the query beyond its original form by incorporating additional context. This is achieved by instructing a LLM to generate a

set (denoted N) of informative passages that capture the underlying intent and context of the user's query.

Similar to the approach [117], we then encode these generated passages using a pretrained encoder to obtain a dense vector representation for the enhanced query. This representation is calculated by averaging the encoded vectors of each individual passage within the set N . By aggregating the representations of the generated passages, we can obtain a more comprehensive query representation for the retrieval process. This enriched representation is used as the query in order to retrieve the k most relevant documents from the collection.

Relevance Assessment

Following recent work [115] that explores the potential of LLMs in improving retrieval quality through relevance assessments, we introduce a straightforward LLM-based relevance assessment mechanism.

Given the initial document ranking, we aim to select a subset of documents while preserving their relative order. However, our primary objective is to guarantee that documents containing the correct answer are included and prioritized within this filtered set. To achieve this, we use a LLM to generate a binary relevance judgment ("yes" or "no") for each document. In simpler terms, for a given query, we instruct the LLM to determine if each document within the top retrieved documents can potentially provide an answer to the query.

Furthermore, the LLM generates relevance judgments sequentially, meaning each document is assessed independently. The LLM receives the query concatenated with the document itself as input for its judgment. Only the top- m documents with a positive ("yes") relevance judgment progress to the next stage.

Rank Aggregation

With the help of passage generation and relevance assessment, we obtain a refined document set, containing a selection of highly relevant documents. Our method then performs a two-stage ranking aggregation process to improve the overall retrieval accuracy. The first stage involves computing individual rankings for each document within the refined set. In the second stage, we aggregate these individual rankings into a single, more robust final ranking. In this way, we aim to improve the diversity of the rankings and reduce the impact of documents incorrectly placed at high rank positions by an individual ranker [118].

Implementation

To promote code reproducibility and transparency, we implemented our system and the baseline retrieval methods using PyTorch²³, a popular deep learning framework. We further used PyFlagr²⁴, a library specifically designed for rank aggregation techniques. Furthermore, we opted for open-source LLMs that are publicly available through Huggingface²⁵. Specifically, we conducted experiments using Solar²⁶ and Mistral²⁷. Our implementation utilizes the pre-built BM25 and Contriever indexes from the Pyserini²⁸ toolkit.

²³ <https://pytorch.org/>

²⁴ <https://flagr.site/>

²⁵ <https://huggingface.co/>

²⁶ <https://huggingface.co/upstage/SOLAR-10.7B-Instruct-v1.0>

²⁷ <https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2>

²⁸ <https://github.com/castorini/pyserini/>

Our experiments were conducted on a workstation equipped with two high-performance Nvidia RTX A6000 GPUs, each with 48GB of memory, and a AMD Ryzen Threadripper PRO 3955WX CPU.

6.2.2 Experiments

Summarizing Crisis Events

We assessed the performance of our approach on the CrisisFACTS 2022 task²⁹. This task focuses on generating daily summaries for short-term crisis events by consolidating factual information from social media and news data relevant to pre-defined queries. Given a set of queries and documents, the objective is to return a list of k most relevant text snippets ("facts") alongside their importance scores.

The CrisisFACTS dataset provides multi-stream data with ground truth summaries sourced from trusted sources including ICS-2009, Wikipedia, and NIST. We used Rouge-2 F1-Score and BERT-Score metrics to evaluate the effectiveness of our approach. We compared our method against the top-performing systems from the CrisisFACTS 2022 challenge, namely, "*unicamp*" and "*ohmkiz*".

System configuration

Within our system configuration, Contriever was utilized for the initial document retrieval stage. The Solar LLM was used to generate ten candidate passages per query for query expansion. The LLM-based relevance assessment was configured to consider the top-5 candidate documents. Finally, a linear rank aggregation technique was employed to merge the individual rankings.

Results

Our approach achieves competitive performance on the CrisisFACTS dataset, as shown in **Table 11**. BERT-Score and ROUGE-F1 metrics indicate fluency and factual accuracy comparable to the best existing methods. This strong performance is consistent across all ground truth summaries, even surpassing all other methods in some cases.

Unlike existing methods like *ohmkiz* that require fine-tuning on question-document pairs or *unicamp* which relies on non-public OpenAI APIs, our method operates entirely in an unsupervised manner and uses readily available, open-source LLMs. In this way our approach fosters broader adoption and replicability within the research community.

²⁹ <https://crisisfacts.github.io/>

Table 11: Results on the CrisisFACTS 2022 dataset. The best performing result is indicated in bold. The second-best is underlined for comparison

	<i>WIKI</i>		<i>NIST</i>		<i>ICS</i>	
	Bert	Rouge	Bert	Rouge	Bert	Rouge
unicamp	53.2	02.8	55.7	<u>13.3</u>	<u>45.9</u>	05.8
ohmkiz	56.4	03.6	56.4	14.7	45.0	<u>05.1</u>
HyDE	53.2	03.0	53.1	11.1	44.4	04.0
Ours	<u>54.0</u>	<u>03.1</u>	<u>56.1</u>	12.6	46.1	04.5

Passage Ranking

Moving beyond the disaster-related benchmarks, we also assessed the performance of our approach in a number of different retrieval tasks. Specifically, we evaluate our method on the TREC 2019 and 2020 [119] Deep Learning Tracks (DL19 and DL20), and five datasets from the BEIR [120] benchmark (Covid, News, NFCorpus, Signal, and Touche). We directly evaluated the performance of our method on the designated test sets for each dataset. Following established evaluation practices, we report MAP, nDCG@10, Recall@10, and Recall@100 for DL19 and DL20, while nDCG@10 is used for the BEIR datasets.

As we emphasize on zero-shot retrieval, we selected baselines that operate without labeled data. These baselines represent distinct retrieval approaches: BM25 serves as a lexicon-based zero-shot method, and Contriever as a dense retrieval approach. To ensure a comprehensive evaluation, we additionally include HyDE [117], a state-of-the-art LLM-based retrieval technique.

Results

In **Table 12** we can see that our method consistently outperforms the baseline methods on both TREC Deep Learning Tracks (DL19 and DL20) datasets. Notably, our method achieves statistically significant improvements of 0.5 to 7.4 percentage points in MAP and nDCG@10 compared to the LLM-based competitor, HyDE. It is worth mentioning that our approach outperforms HyDE on all metrics except Recall@10 (R@10) on DL20. These findings

highlight the effectiveness of our approach, which leverages the combined power of relevance judgments and rank aggregation to achieve better retrieval performance.

Table 12: Results on DL19 and DL20. The best performing result is indicated in bold. The second best is underlined for comparison

	DL19				DL20			
	MAP	nDCG@10	R@10	R@100	MAP	nDCG@10	R@10	R@100
BM25	30.1	50.6	17.8	45.2	28.6	48.0	16.9	55.8
Contriever	24.0	44.5	14.1	48.9	24.0	42.1	23.1	51.4
HyDE _{mistral}	38.2	54.8	22.5	57.4	33.0	49.5	26.8	59.6
HyDE _{solar}	37.4	55.4	22.3	56.9	32.7	<u>52.8</u>	30.2	62.3
Ours(BM25) _{mistral}	38.4	<u>60.4</u>	<u>23.5</u>	55.4	32.4	52.1	26.5	60.9
Ours(BM25) _{solar}	35.5	57.2	20.3	55.9	<u>34.2</u>	52.0	26.9	62.3
Ours(CR) _{mistral}	<u>39.1</u>	56.6	22.7	<u>57.5</u>	33.8	51.5	<u>29.7</u>	60.4
Ours(CR) _{solar}	42.3	62.8	25.4	58.5	34.7	53.3	28.5	<u>61.4</u>

Similar behaviour is observed on the BEIR benchmark datasets in **Table 13**. When combined with either BM25 or Contriever for initial retrieval, our approach consistently surpasses all other methods across all five datasets. While HyDE occasionally exhibits comparable performance, particularly on the NFCorpus dataset, it often falls behind our method by a significant margin.

Comparing the results of combining our approach with different retrievers, it seems that BM25 leads more consistently to good results. Employing Contriever performs very well in some datasets, but not so well in others.

Table 13: Results on BEIR. Best performing is marked bold. The second-best is underlined for comparison

(nDCG@10)	Covid	News	NFCorpus	Signal	Touche
BM25	59.4	39.5	30.7	33.0	44.2
Contriever	27.3	34.8	31.7	23.3	16.6
HyDE _{mistral}	55.9	34.3	30.8	21.6	14.9
HyDE _{solar}	56.7	35.1	31.3	21.5	15.0
Ours(BM25)_{mistral}	61.2	40.9	31.5	<u>33.4</u>	<u>44.7</u>
Ours(BM25)_{solar}	<u>61.5</u>	<u>41.2</u>	32.3	33.6	45.4
Ours(CR)_{mistral}	58.4	40.7	26.7	17.9	18.3
Ours(CR)_{solar}	63.4	41.4	<u>31.9</u>	18.1	18.9

Furthermore, our experiments revealed that Solar exhibited marginally superior overall performance compared to Mistral. This difference could be attributed to the model size, with Solar consisting of 10 billion parameters compared to Mistral's 7 billion.

7 Progress achieved towards the CREXDATA objectives

This deliverable documented the progress of CREXDATA Work Package 4 for the first half of the project. Work Package 4 focuses on advanced learning and forecasting methods that can be executed at scale. The first task of the Work Package (T4.1) is to develop forecasting techniques that cover longer horizons and reach deeper into the future. Towards this end, we presented an expressive forecasting engine and an (online) optimization method for this engine which can help in determining the proper configuration settings. This will enable us to optimize the engine for deeper horizons. We also presented initial forecasting results for the maritime use case. The goal of the second task (T4.2) is to develop algorithms for guiding large-scale simulations towards desired ends. We presented the simulation scenarios from all three CREXDATA use-cases and a discussion of the methods to be applied for interactively exploring the parameter space of the CREXDATA simulators. Another goal of Work Package 4 is to develop bandwidth-efficient algorithms for federated learning (T4.3). We presented a novel bandwidth-efficient technique for Federated Deep Learning and demonstrated its advantage over previous solutions, as well as its applicability in computer vision, which is of crucial importance for the weather emergency use case. CREXDATA also needs to build a scalable, high-throughput service for the optimized execution of distributed data analytics over the cloud. This is the work of T4.4. We presented an approach which optimizes the execution of arbitrarily many workflows over arbitrarily many devices, under arbitrarily many physical execution options in volatile streaming and network settings. The final task of Work Package 4 (T4.5) is to build a robust multi-lingual system that will be able to support decision making by extracting information from multiple social media. We presented language models developed for monitoring and extracting key information about weather emergencies from social media messages.

8 Acronyms and Abbreviations

Each term should be bulleted with a definition.

Below is an initial list that should be adapted to the given deliverable.

- BO – Bayesian Optimization
- CA – Consortium Agreement
- CEF – Complex Event Forecasting
- CEP – Complex Event Processing
- CER – Complex Event Recognition
- D – deliverable
- DoA – Description of Action (Annex 1 of the Grant Agreement)
- EB – Executive Board
- EC – European Commission
- ESC - Exhaustive Search with Counting
- GA – General Assembly / Grant Agreement
- GoG - Graph of Graphs
- GSP - Greedy Search Progressive
- HPC - High Performance Computing
- MCC – Mathews Correlation Coefficient
- RSS - Random Sampling Search
- SREMO – Symbolic Register Expression with Memory and Output
- SRT – Symbolic Register Transducer
- UDF - User Defined Function
- WP – Work Package
- WPL – Work Package Leader

9 References

- [1] E. Alevizos, A. Artikis and G. Paliouras, “Wayeb: a Tool for Complex Event Forecasting,” in *LPAR*, 2018.
- [2] E. Alevizos, A. Artikis and G. Paliouras, “Complex event forecasting with prediction suffix trees,” *VLDB J.*, vol. 31, pp. 157-180, 2022.
- [3] G. Fikioris, K. Patroumpas, A. Artikis, G. Paliouras and M. Pitsikalis, “Fine-Tuned Compressed Representations of Vessel Trajectories,” in *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, 2020.
- [4] S. Pandey, S. Nepal and S. Chen, “A test-bed for the evaluation of business process prediction techniques,” in *CollaborateCom*, 2011.
- [5] V. Muthusamy, H. Liu and H.-A. Jacobsen, “Predictive publish/subscribe matching,” in *DEBS*, 2010.
- [6] V. Stavropoulos, E. Alevizos, N. Giatrakos and A. Artikis, “Optimizing complex event forecasting,” in *Proceedings of the 16th ACM International Conference on Distributed and Event-Based Systems*, New York, NY, USA, 2022.
- [7] “Apache Kafka v. 3.3,” [Online]. Available: <https://kafka.apache.org/>.
- [8] A. Povzner, P. Mahajan, J. Gustafson, J. Rao, I. Juma, F. Min, S. Sridharan, N. Bhatia, G. K. Attaluri, A. Chandra, S. Kozlovski, R. Sivaram, L. Bradstreet, B. Barrett, D. Shah, D. Jacot, D. Arthur, M. Chawla, R. Dagostino, C. McCabe, M. R. Obili, K. Prakasam, J. G. Sancio, V. Singh, A. Nikhil and K. Gupta, “Kora: A Cloud-Native Event Streaming Platform for Kafka,” *Proc. VLDB Endow.*, vol. 16, p. 3822–3834, 2023.
- [9] J. Kreps, N. Narkhede, J. Rao and others, “Kafka: A distributed messaging system for log processing,” in *Proceedings of the NetDB*, 2011.
- [10] N. Giatrakos, E. Alevizos, A. Artikis, A. Deligiannakis and M. N. Garofalakis, “Complex event recognition in the Big Data era: a survey,” *VLDB J.*, vol. 29, pp. 313-352, 2020.
- [11] D. Ron, Y. Singer and N. Tishby, “The Power of Amnesia: Learning Probabilistic Automata with Variable Memory Length,” *Machine Learning*, vol. 25, p. 117–149, 1996.
- [12] D. Ron, Y. Singer and N. Tishby, “The Power of Amnesia,” in *NIPS*, 1993.
- [13] E. Brochu, V. M. Cora and N. de Freitas, *A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning*, 2010.
- [14] P. I. Frazier, *A Tutorial on Bayesian Optimization*, 2018.

- [15] C. Ray, R. Dréo, E. Camossi, A.-L. Jousselme and C. Iphar, "Heterogeneous integrated dataset for Maritime Intelligence, surveillance, and reconnaissance," *Data in Brief*, vol. 25, p. 104141, 2019.
- [16] T. Head, M. Kumar, H. Nahrstaedt, G. Louppe and I. Shcherbatyi, *scikit-optimize/scikit-optimize*, Zenodo, 2021.
- [17] A. J. Demers, J. Gehrke, B. Panda, M. Riedewald, V. Sharma and W. M. White, "Cayuga: A General Purpose Event Monitoring System," in *CIDR*, 2007.
- [18] J. Agrawal, Y. Diao, D. Gyllstrom and N. Immerman, "Efficient pattern matching over event streams," in *SIGMOD Conference*, 2008.
- [19] G. Cugola and A. Margara, "Processing flows of information: From data stream to complex event processing," *ACM Comput. Surv.*, vol. 44, pp. 15:1--15:62, 2012.
- [20] M. Kaminski and N. Francez, "Finite-Memory Automata," *Theor. Comput. Sci.*, vol. 134, pp. 329-363, 1994.
- [21] L. D'Ántoni and M. Veanes, "The Power of Symbolic Automata and Transducers," in *CAV (1)*, 2017.
- [22] A. Grez, C. Riveros and M. Ugarte, "A Formal Framework for Complex Event Processing," in *ICDT*, 2019.
- [23] L. Libkin, T. Tan and D. Vrgoc, "Regular expressions for data words," *J. Comput. Syst. Sci.*, vol. 81, pp. 1278-1297, 2015.
- [24] M. Bucchi, A. Grez, A. Quintana, C. Riveros and S. Vansummeren, "CORE: a COMplex event Recognition Engine," *Proc. VLDB Endow.*, vol. 15, pp. 1951-1964, 2022.
- [25] "SASE Open Source System," [Online]. Available: <https://github.com/haopeng/sase>.
- [26] "Esper," [Online]. Available: <https://www.espertech.com/esper/>.
- [27] "FlinkCEP - Complex event processing for Flink," [Online]. Available: <https://nightlies.apache.org/flink/flink-docs-release-1.17/docs/libs/cep/>.
- [28] M. Pitsikalis, A. Artikis, R. Dreio, C. Ray, E. Camossi and A.-L. Jousselme, "Composite Event Recognition for Maritime Monitoring," in *DEBS*, 2019.
- [29] M. Körber, N. Glombiewski, A. Morgen and B. Seeger, "TPStream: low-latency and high-throughput temporal pattern matching on event streams," *Distributed Parallel Databases*, vol. 39, pp. 361-412, 2021.
- [30] A. Awad, R. Tommasini, S. Langhi, M. Kamel, E. D. Valle and S. Sakr, "D²IA: User-defined interval analytics on distributed streams," *Inf. Syst.*, vol. 104, p. 101679, 2022.
- [31] J. F. Allen, "Towards a General Theory of Action and Time," *Artif. Intell.*, vol. 23, pp. 123-154, 1984.

- [32] R. A. Kowalski and M. J. Sergot, "A Logic-based Calculus of Events," *New Gener. Comput.*, vol. 4, pp. 67-95, 1986.
- [33] A. Artikis, M. J. Sergot and G. Paliouras, "An Event Calculus for Event Recognition," *IEEE Trans. Knowl. Data Eng.*, vol. 27, pp. 895-908, 2015.
- [34] P. Mantenoglou, D. Kelesis and A. Artikis, "Complex Event Recognition with Allen Relations," in *KR*, 2023.
- [35] G. Grigoropoulos, G. Spiliopoulos, I. Chamatidis, M. Kaliorakis, A. Troupiotis-Kapeliaris, M. Voudas, E. Filippou, E. Chondrodima, N. Pelekis, Y. Theodoridis, D. Zissis and K. Bereta, "A Scalable System for Maritime Route and Event Forecasting," in *27th International Conference on Extending Database Technology (EDBT)*, Paestum, Italy, 2024.
- [36] M. Werling, J. Ziegler, S. Kammel and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2010.
- [37] L. Zhao and R. Myung-II, "COLREGs-compliant multiship collision avoidance based on deep reinforcement learning," *Ocean Engineering*, vol. 191, p. 106436, 2019.
- [38] Uber Technologies, "Uber H3: Uber's Hexagonal Hierarchical Spatial Index," 2019.
- [39] M. Ponce-de-Leon, A. Montagud, V. Noël, A. Meert, G. Pradas, E. Barillot, L. Calzone and A. Valencia, "PhysiBoSS 2.0: a sustainable integration of stochastic Boolean and agent-based modelling frameworks," *npj Systems Biology and Applications*, p. 9:54, 2023.
- [40] M. Vázquez, G. Houzeaux, S. Koric, A. Artigues, J. Aguado-Sierra, R. Arís, D. Mira, H. Calmet, F. Cucchiatti, H. Owen, A. Taha, E. D. Burness, J. M. Cela and M. Valero, "Alya: Multiphysics engineering simulation toward exascale," *Journal of Computational Science*, pp. 15-27, 2016.
- [41] C. Akasiadis, M. Ponce-de-Leon, A. Montagud, E. Michelioudakis, A. Atsidakou, E. Alevizos, A. Artikis, A. Valencia and G. Paliouras, "Parallel model exploration for tumor treatment simulations," *Computational Intelligence*, vol. 38, no. 4, pp. 1379-1401, 2022.
- [42] M. Ponce-de-Leon, A. Montagud, C. Akasiadis, J. Schreiber, T. Ntiniakou and A. Valencia, "Optimizing Dosage-Specific Treatments in a Multi-Scale Model of a Tumor Growth," *Frontiers in Molecular Biosciences*, vol. 9, no. 2296-889X, 2022.
- [43] J. Ozik, N. Collier, R. Heiland, G. An and P. Macklin, "Learning-accelerated discovery of immune-tumour interactions," *Molecular systems design & engineering*, vol. 4, no. 4, pp. 747-760, 2019.
- [44] V. Stavropoulos, E. Michelioudakis, C. Akasiadis and A. Artikis, "Resource-effective exploration of tumor treatments with multi-scale simulations," in *Proceedings of the 12th Hellenic Conference on Artificial Intelligence*, Corfu, 2022.

- [45] C. Akasiadis, E. Kladis, P.-F. Kamberi, E. Michelioudakis, E. Alevizos and A. Artikis, “A Framework to Evaluate Early Time-Series Classification Algorithms,” in *EDBT 2024*, Paestum, 2024.
- [46] V. Stavropoulos, E. Alevizos, N. Giatrakos and A. Artikis, “Optimizing Complex Event Forecasting,” in *DEBS '22*, Copenhagen, 2022.
- [47] H. B. McMahan, E. Moore, D. Ramage and B. A. y Arcas, “Federated Learning of Deep Networks using Model Averaging,” *CoRR*, vol. abs/1602.05629, 2016.
- [48] T. Qin, S. R. Etesami and C. A. Uribe, “The role of local steps in local SGD,” *Optimization Methods and Software*, p. 1–27, August 2023.
- [49] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu and X. Zheng, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015.
- [50] S. Li, Y. Zhao, R. Varma, O. Salpekar, P. Noordhuis, T. Li, A. Paszke, J. Smith, B. Vaughan, P. Damania and S. Chintala, “PyTorch Distributed: Experiences on Accelerating Data Parallel Training,” *CoRR*, vol. abs/2006.15704, 2020.
- [51] P. Moritz, R. Nishihara, I. Stoica and M. I. Jordan, *SparkNet: Training Deep Networks in Spark*, 2016.
- [52] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, T. Parcollet and N. D. Lane, “Flower: A Friendly Federated Learning Research Framework,” *CoRR*, vol. abs/2007.14390, 2020.
- [53] L. G. Valiant, “A Bridging Model for Parallel Computation,” *Commun. ACM*, vol. 33, p. 103–111, August 1990.
- [54] O. Dekel, R. Gilad-Bachrach, O. Shamir and L. Xiao, “Optimal Distributed Online Prediction using Mini-Batches,” *CoRR*, vol. abs/1012.1367, 2010.
- [55] M. Zinkevich, M. Weimer, L. Li and A. Smola, “Parallelized Stochastic Gradient Descent,” in *Advances in Neural Information Processing Systems*, 2010.
- [56] S. Shi and X. Chu, “Performance Modeling and Evaluation of Distributed Deep Learning Frameworks on GPUs,” *CoRR*, vol. abs/1711.05979, 2017.
- [57] Y. Lin, S. Han, H. Mao, Y. Wang and W. J. Dally, “Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training,” *CoRR*, vol. abs/1712.01887, 2017.
- [58] S. Shi, Z. Tang, X. Chu, C. Liu, W. Wang and B. Li, “A Quantitative Survey of Communication Optimizations in Distributed Deep Learning,” *IEEE Network*, vol. 35, pp. 230-237, 2021.

- [59] Z. Tang, S. Shi, B. Li and X. Chu, "GossipFL: A Decentralized Federated Learning Framework With Sparsified and Adaptive Communication," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, pp. 909-922, 2023.
- [60] J. Wang and G. Joshi, "Cooperative SGD: A Unified Framework for the Design and Analysis of Local-Update SGD Algorithms," *Journal of Machine Learning Research*, vol. 22, p. 1–50, 2021.
- [61] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar and H. B. McMahan, *Adaptive Federated Optimization*, 2021.
- [62] H. Q. a. M. B. Tzu-Ming Harry Hsu, *Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification*, 2019.
- [63] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. a. Ranzato, A. Senior, P. Tucker, K. Yang, Q. Le and A. Ng, "Large Scale Distributed Deep Networks," in *Advances in Neural Information Processing Systems*, 2012.
- [64] W. Y. S. W. a. Z. Z. Xiang Li, *Communication-Efficient Local Decentralized SGD Methods.*, 2021.
- [65] G. L. M. W. K. F. A.-O. B. a. P. P. Luo Mai, "KungFu: Making Training in Distributed Machine Learning Adaptive," 2020.
- [66] I. J. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, Cambridge, MA: MIT Press, 2016.
- [67] G. Cormode and M. Garofalakis, "Sketching Streams through the Net: Distributed Approximate Query Tracking," in *Proceedings of the 31st International Conference on Very Large Data Bases*, Trondheim, 2005.
- [68] G. Frangias, *Federated learning at TensorFlow Using the geometric approach*, Technical University of Crete, 2023.
- [69] M. Theologitis, *Algorithms for online federated machine learning*, Technical University of Crete, 2023.
- [70] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, p. 141–142, 2012.
- [71] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," p. 32–33, 2009.
- [72] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278-2324, 1998.
- [73] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv e-prints*, p. arXiv:1409.1556, September 2014.
- [74] J. D. H. S. J. K. S. S. S. M. Z. H. A. K. A. K. M. B. A. C. B. a. L. F.-F. Olga Russakovsky, *ImageNet Large Scale Visual Recognition Challenge.*, 2015.

- [75] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Chia Laguna Resort, Sardinia, Italy, 2010.
- [76] G. Huang, Z. Liu and K. Q. Weinberger, "Densely Connected Convolutional Networks," *CoRR*, vol. abs/1608.06993, 2016.
- [77] F. Chollet and others, *Keras*, 2015.
- [78] K. He, X. Zhang, S. Ren and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," *CoRR*, vol. abs/1502.01852, 2015.
- [79] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, L. Zhang, W. Han, M. Huang, Q. Jin, Y. Lan, Y. Liu, Z. Liu, Z. Lu, X. Qiu, R. Song, J. Tang, J.-R. Wen, J. Yuan, W. X. Zhao and J. Zhu, "Pre-Trained Models: Past, Present and Future," *CoRR*, vol. abs/2106.07139, 2021.
- [80] T. Ridnik, E. B. Baruch, A. Noy and L. Zelnik-Manor, "ImageNet-21K Pretraining for the Masses," *CoRR*, vol. abs/2104.10972, 2021.
- [81] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *CoRR*, vol. abs/2010.11929, 2020.
- [82] J. Yosinski, J. Clune, Y. Bengio and H. Lipson, "How transferable are features in deep neural networks?," *CoRR*, vol. abs/1411.1792, 2014.
- [83] D. & B. J. Kingma, Adam: A Method for Stochastic Optimization, International Conference on Learning Representations, 2014.
- [84] I. Sutskever, J. Martens, G. Dahl and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, 2013.
- [85] I. L. a. F. Hutter., "Fixing Weight Decay Regularization in Adam," 2017.
- [86] H. M. C.-Y. W. C. F. T. D. a. S. X. Zhuang Liu, "A ConvNet for the 2020s," 2022.
- [87] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, p. 3021, 2021.
- [88] N. C. Thompson, K. Greenewald, K. Lee and G. F. Manso, "Deep Learning's Diminishing Returns: The Cost of Improvement is Becoming Unsustainable," *IEEE Spectrum*, vol. 58, pp. 50-55, 2021.
- [89] J. L. Schönberger and J.-M. Frahm, "Structure-from-Motion Revisited," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [90] C. C. R. S. & M. D. Paul-Edouard Sarlin, "From Coarse to Fine: Robust Hierarchical Localization at Large Scale.," 2019.

- [91] P. Lindenberg, P.-E. Sarlin, V. Larsson and M. Pollefeys, "Pixel-Perfect Structure-from-Motion with Featuremetric Refinement," *CoRR*, vol. abs/2108.08291, 2021.
- [92] M. W. E. N. E. L. R. Y. J. W. T. K. A. A. J. S. K. A. A. M. D. & K. A. Tancik, "Nerfstudio: A Modular Framework for Neural Radiance Field Development.," *ACM SIGGRAPH 2023 Conference Proceedings.* , 2023.
- [93] N. Giatrakos, E. Alevizos, A. Deligiannakis, R. Klinkenberg and A. Artikis, "Proactive Streaming Analytics at Scale: A Journey from the State-of-the-art to a Production Platform," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023.
- [94] N. Giatrakos, A. Deligiannakis, M. Garofalakis and Y. Kotidis, "Omnibus outlier detection in sensor networks using windowed locality sensitive hashing," *Future Generation Computer Systems*, vol. 110, pp. 587-609, 2020.
- [95] M. Tzortzi, C. Kleitsikas, A. Politis, S. Niarchos, K. Doka and N. Koziris, "Planning Workflow Executions over the Edge-to-Cloud Continuum," in *International Symposium on Algorithmic Aspects of Cloud Computing*, 2023.
- [96] S. Zeuch, X. Chatziliadis, A. Chaudhary, D. Giouroukis, P. M. Grulich, D. P. A. Nugroho, A. Ziehn and V. Mark, "NebulaStream: Data Management for the Internet of Things," *Datenbank-Spektrum*, vol. 22, pp. 131-141, 2022.
- [97] M. N. L. Carvalho, A. Queralt, O. Romero, A. Simitsis, C. Tatu and R. M. Badia, "Performance Analysis of Distributed GPU-Accelerated Task-Based Workflows," 2024.
- [98] K. Chasialis, T. Palaiologou, Y. Foufoulas, A. Simitsis and Y. Ioannidis, "QFusor: A UDF Optimizer Plugin for SQL Databases," 2024.
- [99] H. Herodotou, Y. Chen and J. Lu, "A survey on automatic parameter tuning for big data processing systems," *ACM Computing Surveys (CSUR)*, vol. 53, pp. 1-37, 2020.
- [100] D. Tsesmelis and A. Simitsis, "Database optimizers in the era of learning," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2022.
- [101] Z. Kaoudi, J.-A. Quiané-Ruiz, B. Contreras-Rojas, R. Pardo-Meza, A. Troudi and S. Chawla, "ML-based cross-platform query optimization," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, 2020.
- [102] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," *Software: Practice and Experience*, vol. 47, pp. 1275-1296, 2017.
- [103] C. Sonmez, A. Ozgovde and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," *Transactions on Emerging Telecommunications Technologies*, vol. 29, p. e3493, 2018.
- [104] C. Mechalik, H. Taktak and F. Moussa, "PureEdgeSim: A simulation toolkit for performance evaluation of cloud, fog, and pure edge computing environments," in

- 2019 international conference on high performance computing & simulation (HPCS)*, 2019.
- [105] M. Salama, Y. Elkhatab and G. Blair, "IoTNetSim: A modelling and simulation platform for end-to-end IoT services and networking," in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, 2019.
- [106] D. N. Jha, K. Alwasel, A. Alshoshan, X. Huang, R. K. Naha, S. K. Battula, S. Garg, D. Puthal, P. James, A. Zomaya and others, "IoTsim-Edge: a simulation framework for modeling the behavior of Internet of Things and edge computing environments," *Software: Practice and Experience*, vol. 50, pp. 844-867, 2020.
- [107] M. Barika, S. Garg, A. Chan, R. N. Calheiros and R. Ranjan, "IoTsim-Stream: Modelling stream graph application in cloud simulation," *Future Generation Computer Systems*, vol. 99, pp. 86-105, 2019.
- [108] X. Zeng, S. K. Garg, P. Strazdins, P. P. Jayaraman, D. Georgakopoulos and R. Ranjan, "IOTSim: A simulator for analysing IoT applications," *Journal of Systems Architecture*, vol. 72, pp. 93-107, 2017.
- [109] A. Shukla, S. Chaturvedi and Y. Simmhan, "Riotbench: An iot benchmark for distributed stream processing systems," *Concurrency and Computation: Practice and Experience*, vol. 29, p. e4257, 2017.
- [110] F. Ofli, F. Alam and M. Imran, "Analysis of social media data using multimodal deep learning for disaster response," *arXiv preprint arXiv:2004.11838*, 2020.
- [111] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
- [112] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. I. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier and others, "Mistral 7B," *arXiv preprint arXiv:2310.06825*, 2023.
- [113] Y. Chae and T. Davidson, "Large language models for text classification: From zero-shot learning to fine-tuning," *Open Science Foundation*, 2023.
- [114] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.
- [115] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar and others, "Holistic evaluation of language models," *arXiv preprint arXiv:2211.09110*, 2022.
- [116] P. Thomas, S. Spielman, N. Craswell and B. Mitra, "Large language models can accurately predict searcher preferences," *arXiv preprint arXiv:2309.10621*, 2023.
- [117] L. Gao, X. Ma, J. Lin and J. Callan, "Precise zero-shot dense retrieval without relevance labels," *arXiv preprint arXiv:2212.10496*, 2022.

- [118] J. Alcaraz, M. Landete and J. F. Monge, “Rank Aggregation: Models and Algorithms,” in *The palgrave handbook of operations research*, Springer, 2022, pp. 153-178.
- [119] N. Craswell, B. Mitra, E. Yilmaz, D. Campos and E. M. Voorhees, “Overview of the TREC 2019 deep learning track,” *arXiv preprint arXiv:2003.07820*, 2020.
- [120] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava and I. Gurevych, “Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models,” *arXiv preprint arXiv:2104.08663*, 2021.
- [121] M. Feurer, J. Springenberg and F. Hutter, “Initializing Bayesian Hyperparameter Optimization via Meta-Learning,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, February 2015.
- [122] A. Artikis, N. Katzouris, I. Correia, C. Baber, N. Morar, I. Skarbovsky, F. Fournier and G. Paliouras, “A Prototype for Credit Card Fraud Management: Industry Paper,” in *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems, DEBS 2017, Barcelona, Spain, June 19-23, 2017*, 2017.

10 Appendix 1

10.1 Appendix for Online Optimization of Complex Event Forecasting

10.1.1 RTCEF services

We present with more details the processing of each service comprising our framework. Synchronisation of the various services is denoted by dotted arrows in Figure 3.

Collector. As mentioned earlier, the collector service is responsible for organising and storing up-to-date datasets from the input stream. Therefore, the collector service consumes the input stream (see ‘Data collection’ in **Figure 3**), in parallel to the forecasting engine, and stores subsets of it in time buckets of fixed `bucket_size`. The collector collects data in a sliding window manner, i.e., it creates a dataset version using the last `dt_size` buckets from the current time. A new dataset version, containing the bucket range (e.g., `[Bucket56, Bucket59]`), is created and emitted to the ‘datasets’ topic as soon as the last bucket in the range is full. Old buckets that no longer serve a purpose, i.e. they are not part of a dataset to be or being used by the factory service, are deleted for space economy.

Forecasting Engine. The forecasting engine, as illustrated at the top part of **Figure 3**, consumes the input stream and produces CE forecasts as well as forecast performance metrics. The distance between two consecutive reports is controlled by the `reports_distance` parameter. Each performance report concerns the last batch of the input stream since the previous report—this batch has equal size to the `reports_distance` parameter. Report contents include, the number of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN), as well as the MCC value and a timestamp of the report. Reports are fed into the ‘reports’ topic and subsequently consumed by the ‘observer’ service. In addition to the above, Wayeb monitors the ‘Models’ topic for new model versions. When a new model is available, Wayeb replaces its model with the latest available version.

Observer. The observer service monitors the performance of the forecasting engine and produces ‘retrain’ or ‘optimise’ instructions as indicated by the policy of **Algorithm**, which works as follows. The observer, consumes a new score from the reports topic (see top right of **Figure 3**) and retains up to k recent scores (lines 4-5)—in our case each score is an MCC value. Then the observer will compute the first degree polynomial $f_c(x) = a_c x + b_c$ (a trend line) so that a_c and b_c minimize the following squared error $E = \sum_{j=0}^{j=k} |f(x_j) - y_j|$ for $x_j = j$ and $y_j = score_{c-k+j}$, where c is an increasing integer denoting the id of the current score (lines 9,10). If the slope (a_c) of $f_c(x)$ is negative (indicating decrease in performance) and less than a `max_slope` $\in \mathbb{R}^+$ parameter (line 11) then a ‘retrain’ instruction is produced and fed into the ‘instructions’ topic (lines 15-17). Intuitively, forecasting performance deterioration shortly after a new model deployment (i.e., $a_c < \text{max_slope}$), indicates that the model failed, and optimisation should thus be performed. Consequently, we place each newly deployed model in a guard period (lines 14,17). A guard period starts after a model is deployed, and ends after `guard_n` performance reports. If the performance of a model under a guard period deteriorates ($a_c < \text{min_slope}$) then an optimisation instruction is produced and fed into the ‘instructions’ topic (lines 12,13). If on the other hand, $a_c < \text{min_slope}$ is satisfied after `guard_n` reports, then a ‘retrain’ instruction is produced for which a new “guard” period begins. Finally, to avoid pitfalls whereby the score drops suddenly very low, we employ an

additional condition: if the score of a report is lower than a threshold min_score (line 6) then the observer asks directly for ‘optimisation’ and omits a ‘retrain’ instruction.

Algorithm 5: Observer service

Require: $k, \text{guard}_n, \text{max_slope}, \text{min_score}$

```

1:  $\text{scores} \leftarrow []$ 
2:  $\text{guard} \leftarrow -1$ 
3: while True do
4:    $\text{score}_c \leftarrow \text{consume}(\text{reports})$ 
5:    $\text{scores.update}(\text{score}_c, k)$ 
6:    $\text{pit\_cond} \leftarrow \text{score}_c < \text{min\_score}$ 
7:    $\text{slope\_cond} \leftarrow \text{False}$ 
8:   if  $\text{guard} \geq 0$  then  $\text{guard} \leftarrow \text{guard} - 1$ 
9:   if  $|\text{scores}| > 2$  then
10:     $(a_c, b_c) \leftarrow \text{fit\_trend}(\text{scores})$ 
11:     $\text{slope\_cond} \leftarrow a_c < \text{max\_slope}$ 
12:   if  $(\text{slope\_cond} \text{ and } \text{guard} \geq 0)$  or  $\text{pit\_cond}$  then
13:     send("instructions", "optimise")
14:      $\text{guard} \leftarrow \text{guard}_n$  ▷ New guard period
15:   else if  $\text{slope\_cond}$  then
16:     send("instructions", "retrain")
17:      $\text{guard} \leftarrow \text{guard}_n$  ▷ New guard period

```

Controller. The controller service controls model update procedures. To do so, the controller, reads messages from the ‘instructions’ topic containing either ‘retrain’ or ‘optimisation’ instructions. Then, accordingly, it requests the model factory service to produce a model via retraining, or initialises an optimisation procedure.

Retrain. In this case the controller sends a ‘train’ command along with the currently best hyper-parameters. These hyper-parameters are either retrieved from the last optimisation procedure, or provided by users if no optimisation has been performed yet.

Optimisation. Bayesian Optimisation (BO) requires a few samples obtained through micro-benchmarks (see Background). Therefore when optimisation is required, the controller service initiates a message exchange that works as follows. First, the controller sets up the optimiser. Similar to [121], we leverage micro-benchmarks from previous runs. More specifically, using the $\text{retain_percentage} \in [0,1]$ parameter, we uniformly keep $[\text{retain_percentage} * \text{all_samples}]$ observations from the last executed BO run, where all_samples is the total number of micro-benchmarks. This allows us to skip the initial sampling step of BO, accelerate optimisation, and retain useful information from previous runs. In addition to the above, the controller sends to the factory an initialisation message so that the appropriate resources are reserved. Then, the controller and the factory enter into the ‘step phase’. During the step phase the controller sends ‘train & test’ commands to the factory along with the hyper-parameters suggested by the acquisition function. After each ‘train & test’ command the controller awaits the corresponding performance report i.e., the value of the $\text{Score}(c)$ objective function. Upon receiving the performance report, the optimiser is updated with the new sample and the next hyper-parameters are suggested for the next step. The step phase ends when all micro-benchmarks are completed. Finally, once all micro-benchmarks are completed and the best hyper-parameters are acquired, the controller sends a finalisation message to the factory containing the ID of the best model. At

the same time, the controller updates the previously best hyper-parameters with the newly acquired ones, ensuring availability of the latter for subsequent 'retrain' instructions.

Model factory. The model factory service trains, tests and sends models to the forecasting engine. Moreover, the factory reads dataset versions from the 'Datasets' topic and assembles temporary datasets using the received bucket ranges (included in a dataset version message). Upon receiving a 'train' command the factory trains a model on the latest assembled dataset and sends a new model version to the 'Models' topic. Concerning production of models through optimisation, the factory first 'locks' the most recent assembled dataset so that the same dataset is used throughout the optimisation procedure. Then it trains, saves and tests candidate models on the locked dataset and reports prediction performance metrics to the controller. Finally, once optimisation is completed, the factory sends a new model version to the 'Models' topic along with its production time. It is only at this point, that the CEF engine will stop momentarily for model replacement.

10.1.2 Financial experiments

Experimental setup

Credit card fraud management. We use a synthetic dataset provided by Feedzai³⁰ containing 1M credit card transaction events taking place over a period of 82 weeks. Each event contains, among others, the card ID, the amount and time of the transaction. We evaluate RTCEF on a pattern, representing a fraudulent behaviour as a sequence of consecutive increasing transactions [122]:

$$R_{\text{cards}} := (\text{amountDiff} > 0) \cdot (\text{amountDiff} > 0) \cdot (\text{amountDiff} > 0) \cdot \\ (\text{amountDiff} > 0) \cdot (\text{amountDiff} > 0) \cdot (\text{amountDiff} > 0) \cdot \\ (\text{amountDiff} > 0)$$

We enrich events of the input stream with an additional attribute amountDiff which is equal to the difference between the previous transaction and the current one. Therefore R_{cards} is satisfied when 8 consecutive transactions happen with increasing amounts. In order to simulate evolving fraud, we modify the financial dataset by changing randomly every 4 to 8 weeks the range of the highly correlated feature amountDiff. Similar to the maritime dataset, for validating our results, we create 21 datasets (FD_i , $i \in [0,20]$) by shifting 4 weeks the start of each dataset in a cyclic manner.

Experimental results

Financial fraud management. Figure 48 left, shows that for FD_0 RTCEF overcomes input data evolutions, and significantly outperforms the offline approach. A similar pattern can be observed also for dataset FD_8 (Figure 48 right), where changes in the input drop the MCC score of the offline approach to almost 0. Again in this case, RTCEF, adapts and maintains overall a steady MCC over time. An interesting experiment, is that of dataset FD_1 (Figure 48 middle). Here, after optimisation is requested on week 26, RTCEF produces steadily an MCC score of ~ 0.75 . Conversely, the offline approach on weeks 35-55 and 61-71 produces a score of ~ 0.85 , thus outperforming RTCEF. In this case, RTCEF fails to detect input stream data evolutions as after week 26 there are no major fluctuations in forecasting performance.

³⁰ <https://feedzai.com>

Figure 49 bottom, shows that for FD_1 our framework has slightly less average MCC , than the offline approach. However, for all other datasets FD_i , $RTCEF$ significantly outperforms the offline approach (see Figure 49).

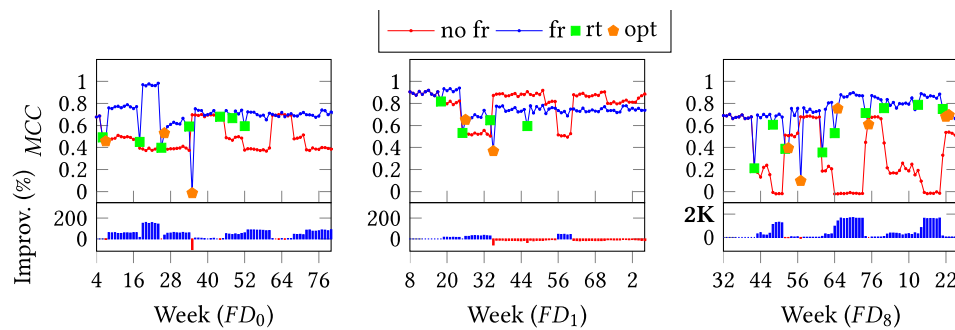


Figure 48: Experimental results for datasets $FD\{0/3/5\}$ with R_{cards} . 'rt' and 'opt' denote 'retrain' and 'optimisation'. Upper plots show MCC over time, while lower plots show improvement

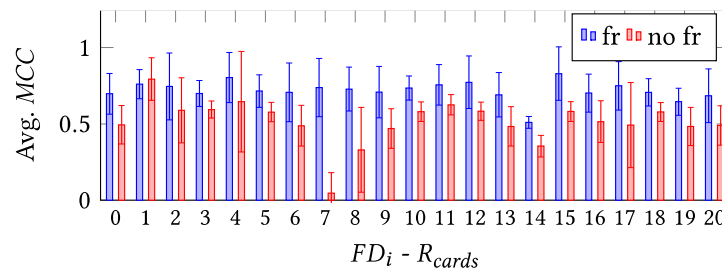


Figure 49: Avg MCC per FD_i for R_{cards}